UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FINAL DE GRADO

Grado en Ingeniería Biomédica

Reconocimiento Automático de Células Malignas en Sangre Periférica a partir de Imágenes Digitales y utilizando Redes Neuronales Convolucionales

Memoria y Anexos

Autor: Rebeca Villarraso Jiménez

Director: José Julian Rodellar Benede

Co-Director: Santiago Alférez Baquero

Convocatoria: 2017-2018

A mis padres, sin los que este trabajo no hubiese sido posible.

I. RESUM

Existeix una gran varietat de patologies que afecten als glòbuls blancs, des de malalties benignes a patologies greus. Les leucèmies son patologies que s'originen per proliferacions incontrolades dels precursors immadurs dels elements sanguinis que circulen a la sang (leucòcits, hematies i plaquetes). Els limfomes son proliferacions incontrolades de la sèrie limfoide als teixits on té lloc la seva maduració (òrgans limfoides). Per al seu diagnòstic es realitza l'anàlisi morfològic de les cèl·lules extretes de la sang perifèrica (vasos sanguinis). Aquesta tècnica per a algunes patologies es imprescindible i requereix de personal qualificat i expert, ja que la morfologia entre els diferents estats de maduració dels glòbuls blancs tenen diferències lleus. La morfologia cel·lular a patologies com els estats febrils mostra canvis que també presenten cèl·lules que proliferen en altres patologies greus com les leucèmies. Per lo tant, es de vital importància per als pacients, realitzar una classificació cel·lular correcta amb el fi de determinar la gravetat patològica i derivar al pacient a l'especialista mèdic adequat en la major brevetat possible. Existeixen equips de classificació cel·lular que preclassifiquen cèl·lules sanguínies, però son molt costosos i funcionen millor amb cèl·lules normals.

L'objectiu d'aquest treball es implementar un algoritme automatitzat que permeti classificar les imatges de les cèl·lules extretes dels vasos sanguinis de forma general, en quatre tipus: limfòcits anormals associats a limfomes, blasts associats a leucèmies, limfòcits normals i limfòcits reactius associats a patologies infeccioses benignes. Per això s'utilitzen eines d'anàlisi profund de xarxes neuronals convolucionals.

Es processa el conjunt de dades proporcionat per professionals del Laboratori Core de l'Hospital Clínic de Barcelona i investigadors del grup CoDAlab. Es tracta d'imatges digitals de sang perifèrica, i es fan servir per entrenar algoritmes amb diferents tècniques: entrenament d'una xarxa neuronal convolucional des de cero, extracció de característiques amb models preentrenats i ajust fi (*Fine Tuning*) de models preentrenats amb transferència d'aprenentatge.

Després de realitzar diferents experiments, s'aconsegueix un algoritme amb efectivitat de classificació de cèl·lules d'un 96,25% amb el model *Xception*, per lo que es pot concloure, que l'ús de xarxes neuronals convolucionals per a la classificació de cèl·lules es satisfactori.

Paraules clau: intel·ligència artificial, xarxes neuronals convolucionals, visió per computador, deep learning, classificació de imatges, blast, limfòcit, leucèmia, limfoma

II. RESUMEN

Existe una gran variedad de patologías que afectan a los glóbulos blancos, desde enfermedades benignas a patologías graves. Las leucemias son patologías que se originan por proliferaciones incontroladas de los precursores inmaduros de los elementos sanguíneos que circulan en la sangre (leucocitos, hematíes y plaquetas). Los linfomas son proliferaciones incontroladas de la serie linfoide en los tejidos donde tiene lugar su maduración (órganos linfoides). Para su diagnóstico se realiza un análisis morfológico de las células extraídas de la sangre periférico (vasos sanguíneos). Esta técnica para algunas patologías es imprescindible y requiere de personal cualificado y experto, ya que la morfología entre los diferentes estados de maduración de los glóbulos blancos tiene diferencias leves. La morfología celular en patologías como los estados febriles muestra cambios que también presentan las células que proliferan en otras patologías graves como las leucemias. Por lo tanto, es de vital importancia para los pacientes, realizar una clasificación celular correcta con el fin de determinar la gravedad patológica y derivar al paciente al especialista médico adecuado en la mayor brevedad posible. Existen equipos que preclasifican células sanguíneas, pero son muy costosos y funcionan mejor con células normales.

El objetivo de este trabajo es implementar un algoritmo automatizado que permita clasificar las imágenes de las células extraídas de los vasos sanguíneos de una forma general, en cuatro tipos de células: linfocitos anormales asociados a linfomas, blastos asociados a leucemias, linfocitos normales y linfocitos reactivos asociados a patologías infecciosas benignas. Para ello se utilizan herramientas de análisis profundo de redes neuronales convolucionales.

Se preprocesa el conjunto de datos proporcionado por los profesionales del Laboratorio Core del Hospital Clínic de Barcelona e investigadores del grupo CoDAlab. Se trata de imágenes digitales de sangre periférica, y se utilizan para entrenar algoritmos con diferentes técnicas: entrenamiento de una red neuronal convolucional desde cero, extracción de características con modelos preentrenados y ajuste fino (*Fine Tuning*) de modelos preentrenados con transferencia de aprendizaje.

Después de los diferentes experimentos, se consigue un algoritmo con efectividad de clasificación de células de un 96,25% con el modelo *Xception*, por lo que se puede concluir, que el uso de redes neuronales convolucionales para clasificación de células es satisfactorio.

Palabras clave: inteligencia artificial, redes neuronales convolucionales, visión por computador, deep learning, clasificación de imágenes, blasto, linfocito, leucemia, linfoma

III. ABSTRACT

Exists a wide variety of pathologies that affect white blood cells, from benign diseases to serious pathologies. Leukemia are pathologies that originate by uncontrolled proliferations of the immature precursors of the blood elements circulating in the blood (leukocytes, red blood cells and platelets). Lymphomas are uncontrolled proliferations of the lymphoid series in the tissues where their maturation takes place (lymphoid organs). For its diagnosis, the morphological analysis of the cells extracted from the peripheral blood (blood vessels) is performed. This technique for some pathologies is essential and requires qualified and expert personnel, since the morphology between the different stages of maturation of white blood cells has slight differences. Cellular morphology in pathologies such as febrile states shows changes that also present cells that proliferate in other serious pathologies such as leukemia. Therefore, it is of vital importance to patients, perform a correct cell classification in order to determine the pathological severity and refer the patient to the appropriate medical specialist as soon as possible. There are cell division devices that pre-classify blood cells, but they are very expensive and work better with normal cells.

The objective of this work is to implement an automated algorithm that allows to classify the images of cells extracted from blood vessels in a general way, in four types of cells: abnormal lymphocytes associated with lymphomas, blasts associated with leukemia, normal lymphocytes and reactive lymphocytes associated with benign infectious pathologies. To this end, deep analysis tools of convolutional neural networks are used.

The set of data provided by professionals from the Laboratory Core of the Hospital Clínic de Barcelona and researchers from the CoDAlab group is processed. These are digital images of peripheral blood, and are used to train algorithms with different techniques: training of a convolutional neural network from scratch, extraction of features with pre-trained models and fine tuning of pre-trained models with learning transfer.

After the different experiments, an algorithm with 96.25% cell classification effectiveness is achieved with the *Xception* model, so it can be concluded that the use of convolutional neural networks for cell classification is satisfactory.

Keywords: artificial intelligent, convolutional neural networks, computer vision, deep Learning, image classification, blasts, lymphocytes, leukemia, lymphoma.

IV. AGRADECIMIENTOS

Quisiera dar mis más sinceros agradecimientos a José Rodellar por entender mis inquietudes desde el inicio del proyecto, y especialmente a Santiago Alférez por su dedicación interminable, por la guía durante todo el proyecto y por la entrega de los recursos necesarios que han podido permitirme comprender las redes neuronales convolucionales de forma efectiva. Quisiera darles las gracias a los dos, no sólo por sus conocimientos, sino por la calidad de su enseñanza.

Agradecer también a la Dra. Anna Merino, del Laboratorio Core del Hospital Clínic de Barcelona y miembro del grupo de investigación CoDAlab, por su valioso apoyo y ayuda con las imágenes para la realización de este proyecto.

V. GLOSARIO

ANN (Artificial Neural Network): red neuronal artificial

CNN (Convolutional Neural Network): red neuronal convolucional

ConvNet: red neuronal convolucional

Deep Learning: aprendizaje profundo con redes neuronales artificiales de gran tamaño

Machine Learning: técnicas aplicadas a algoritmos para aprendizaje automático

INDICE

1. PREFACIO

1.1. Contenido de la memoria

En el presente capítulo se define el contenido de la memoria, el origen del trabajo y su motivación. En el **Capítulo 2** se describe la introducción al trabajo, sus objetivos y su alcance. En el **Capítulo 3**, se desarrollan las bases biológicas del estudio y la metodología a aplicar. Del **Capítulo 4 al 7** se detallan las técnicas aplicadas y sus resultados. En el **Capítulo 8** se realiza el análisis de impacto ambiental del estudio. Finalmente se incluyen las Conclusiones, el Análisis Económico, la Bibliografía consultada y el Anexo.

1.2. Origen del trabajo

El presente trabajo nace de la necesidad de utilizar una metodología automatizada para la clasificación de imágenes de células de sangre proveniente de vasos sanguíneos (sangre periférica) de forma rápida, económica y que no dependa del sistema de adquisición de imágenes. Se basa en la tesis doctoral "*Methodology for Automatic Classification of Atypical Lymphoid Cells from Peripherial Blood Cell Images*", del grupo de investigación CodaLab, presentada por Santiago Alférez Baquero en el año 2015 en el Programa de Ingeniería Biomédica de la Universitat Politècnica de Catalunya *[1]* y dirigida por el Profesor José Rodellar y la Dra. Anna Merino del Hospital Clínic de Barcelona. En el presente trabajo se experimenta con técnicas avanzadas de Inteligencia Artificial y la realización de nuevos algoritmos basados en Redes Neuronales Convolucionales CNN (por sus siglas en inglés) con el fin de determinar el más efectivo para la clasificación de células sanguíneas.

1.3. Motivación

Después de realizar la asignatura optativa "Inteligencia Artificial" durante los estudios de Ingeniería Biomédica, en la que pude conocer el potencial de las técnicas de Machine Learning y a partir de la asignatura "Procesado de Imágenes Biomédicas", apareció en mí el interés de conocer las Redes Neuronales Artificiales ANN (por sus siglas en inglés) y su aplicación en imágenes médicas. Por este motivo, me pareció interesante la aplicación de ANN en el proyecto propuesto por José Rodellar y Santiago Alférez (Director y Codirector de este trabajo).

Me especificaron el problema a resolver y la necesidad de comprender las Redes Neuronales Convolucionales, que son un tipo de ANN aplicadas a Visión por Computador (imágenes). Después de conocer la aplicabilidad de estas técnicas al problema de diferenciación de células sanguíneas en enfermedades graves, surgió en mí mucha más fascinación.

1.4. Necesidades previas

Al disponer de conocimientos muy vagos sobre ANN y ninguna en CNN, se realizó el estudio de varios libros básicos *[4]*, *[5]*, un tutorial *[10]* y la realización de una especialización en Deep Learning que constaba de cinco cursos ofrecidos online por la Universidad de Stanford *[6]*. Además se han utilizado herramientas como Anaconda, librerías TensorFlow y Keras que han requerido la consulta de información técnica para la realización de los algoritmos. Posteriormente surgió la necesidad de la configuración del equipo personal para la conexión remota a la UPC y acceso a la GPU para el entrenamiento de los algoritmos.

Para la comprensión de las necesidades reales del estudio y sus bases biológicas se ha requerido el estudio de libros médicos sobre Hematología *[2]*, Histología *[3]* y Oncología *[13]*.

Todos los experimentos se han realizado a través de una computadora portátil con las siguientes características:

- Equipo: AsusTeK® N71 Series
- Sistema Operativo: Windows © 7 Home Premium 64 bits
- Procesador: Intel® Core™ i5 – 2450M CPU @ 2,50GHz
- Memoria RAM: 4GB
- Tarjeta gráfica: NVIDIA GEFORCE® GT 2GB

El software que se ha requerido instalar para la realización de los algoritmos y entrenamientos en el anterior equipo es el siguiente:

- Python 3.6 *[30]*
- Anaconda Navigator 3-5.1.0 para Python 3.6 y Windows © 64 bits *[25]*
- Librería TensorFlow *[26]*
- Librería Keras *[27]*
- 7-Zip Manager *[29]*

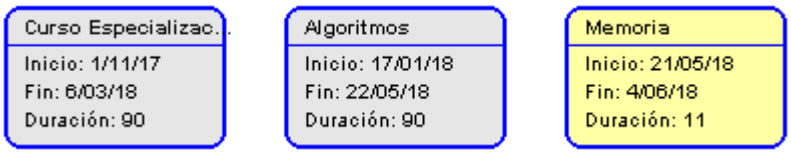
Para la conexión remota a la UPC y acceder al procesador GPU se ha requerido instalar:

- Pulse Secure *[28]*
- Servidor con Deepbox con tarjeta gráfica Nvidia Titan Xp 30 GB disponibles.

El software utilizado es de acceso libre y sin coste.

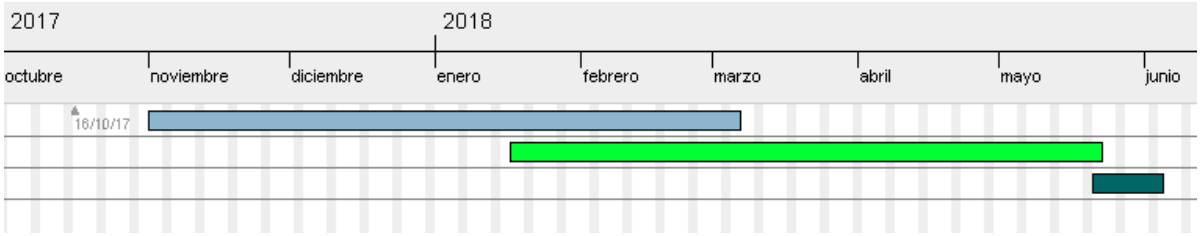
1.5. Planificación

El tiempo empleado por la autora del proyecto es de 24 ECTS académicos a 25 horas cada ECT, que corresponden a 600 horas computables, aunque la formación y las primeras pruebas con algoritmos se iniciaron antes de la matriculación al TFG. Las tareas realizadas se distribuyen en tres grandes bloques:



- Curso de especialización en Deep Learning *[6]*
- Preprocesado del conjunto de datos, implementación de algoritmos y entrenamiento
- Redacción de la memoria, revisiones y rectificaciones.

Diagrama de Gantt



2. INTRODUCCIÓN

2.1. Introducción

Existen diferentes patologías hematológicas benignas, tales como los estados febriles y afecciones víricas que tienen síntomas similares a los iniciales de enfermedades cancerígenas graves como las leucemias (cáncer de las células que maduran en la médula ósea) y linfomas (cáncer de células que maduran en los ganglios linfáticos) *Cap.87-88, [13]*.

Para el diagnóstico de estas enfermedades hematológicas los especialistas sanitarios utilizan, entre otras técnicas, el análisis morfológico de células extraídas de los vasos sanguíneos (sangre periférica). La evaluación de las características morfológicas de células normales, son relativamente fáciles de reconocer. En cambio, las células malignas sobre todo las linfoides, son muy complejas de distinguir, ya que las diferencias morfológicas son leves entre las diferentes clases. *[1]*

Actualmente las tareas de detección y clasificación de células las pueden realizar equipos costosos que trabajan mejor con células sanas o bien, personal cualificado muy experimentado. Con el fin de facilitar el diagnóstico morfológico inicial de estas enfermedades hematológicas, y ayudar a una detección temprana y posibilitar un tratamiento rápido a los pacientes con este tipo de enfermedades, se implementará un método de clasificación celular automatizado para el grupo de enfermedades de origen infeccioso o maligno (leucemias o linfomas), el cual podría ser utilizado como método de soporte diagnóstico, y que únicamente requiere de extracción sanguínea (fácilmente accesible), microscopio simple, cámara digital y computadora con el algoritmo desarrollado.

2.2. Objetivos del trabajo

El **objetivo general** de este trabajo es el desarrollo de un método automatizado de detección y clasificación de células sanguíneas (glóbulos blancos) a partir de imágenes obtenidas de frotis sanguíneo (*Ver Capítulo 3.4*) con microscopio óptico y cámara digital. La clasificación se realizará para cuatro grandes grupos de células: linfocitos normales, linfocitos anormales (asociados a linfomas), linfocitos reactivos (procesos infecciosos) y blastos (asociados a leucemias), lo que permitirá realizar una ayuda al diagnóstico inicial entre sujetos con linfomas, leucemias, procesos infecciosos, o patologías no graves. Cabe destacar que el frotis de un paciente patológico puede incluir células normales y patológicas, por lo que la presencia de linfocitos normales no implica que el paciente esté sano.

Para la consecución del objetivo específico, ha sido necesario el estudio de los siguientes ítems**:**

- Comprender: la composición de la sangre, tipos celulares, su formación y función. Patologías asociadas a los glóbulos blancos. Métodos de detección hematológicos.
- La comprensión de las redes neuronales profundas, especialmente las redes neuronales convolucionales, para su aplicabilidad al desarrollo de nuevos algoritmos. Esto involucra el

aprendizaje de diferentes herramientas como Jupyter, Anaconda y, librerías TensorFlow y Keras.

- Análisis inicial del conjunto de datos, para su tratamiento y preprocesado.
- Aplicar técnicas de balanceado de datos para evitar el sobreajuste.
- Comprensión y aplicación de las diferentes técnicas de regularización, optimización y funciones de activación, así como el ajuste de hiperparámetros para hacer eficientes los algoritmos cuando existen pocos ejemplos en el conjunto de datos.
- Compresión y uso de modelos preentrenados de CNN de acceso público: *VGG16*, *VGG19*, *ResNet50*, *InceptionResNetV1*, *InceptionResNetV2*, *InceptionV3*, *InceptionV4* y *Xception*.

2.3. Alcance del trabajo

Algunos linfocitos anormales asociados a linfomas tienen características morfológicas parecidas con respecto a las células blásticas asociadas a leucemias agudas. Estas dos patologías tienen tratamiento y pronóstico diferente, y su diferenciación temprana aumenta la posibilidad de supervivencia de los afectados. Por otro lado, los linfocitos reactivos asociados a infecciones no malignas, presentan morfología similar a la de los linfocitos anormales característicos de linfomas, por lo que es vital una clasificación correcta. *[1]*

Este trabajo pretende ayudar al diagnóstico inicial entre las patologías mencionadas, con el fin de detectar las cancerígenas graves y derivar al paciente a un centro especializado de forma rápida y económica. Cabe destacar, que dentro de los tres grandes grupos de patologías mencionadas (procesos infecciosos/febriles/víricos, linfomas y leucemias) existen diferentes subclasificaciones de células que permitirían detectar la patología concreta. Por ejemplo, dentro de los linfocitos existen los de tipo B y T, y existen diferentes tipos de linfomas de tipo B y de tipo T con diferentes alteraciones clínicas y de otras pruebas complementarias. Sin embargo, no se distinguen entre subclases, ya que está fuera del alcance del presente trabajo. Por otro lado, se debe tener en cuenta, que el grupo de patologías asociado a los glóbulos blancos es muy amplio y no sólo está asociado a linfomas, leucemias y procesos infecciosos víricos/febriles, ya que, existen otro tipo de patologías que también pueden ser graves, como algunas infecciones por microorganismos que implican disfunciones importantes. Así pues, debe aceptarse la limitación de los algoritmos a una clasificación básica de células para continuar con las pruebas diagnósticas necesarias.

3. BASES BIOLÓGICAS DEL ESTUDIO

3.1. La sangre

La sangre es un líquido viscoso constituido por células y plasma, y circula a través de los vasos sanguíneos. Si se introduce sangre en un tubo de ensayo y se le deja en reposo, se distinguirán tres capas. La inferior, roja, está compuesta por los **glóbulos rojos** o **eritrocitos**; una fina gris formada por **plaquetas** o **trombocitos** y por **glóbulos blancos** o **leucocitos;** y en la superior se distinguirá un líquido translúcido amarillento, el **plasma** sanguíneo. (*Cap. 10, p. 233 [3]*).

Los glóbulos rojos y plaquetas no tienen núcleo y sólo realizan su función dentro de los vasos sanguíneos, en cambio los glóbulos blancos tienen núcleo y están de forma transitoria en el torrente sanguíneo, ya que viajan a diferentes **tejidos** y/o a **órganos linfoides** (*Apartado 3.3*).

3.2. Células sanguíneas y Hematopoyesis

Las células sanguíneas comienzan sus vidas en la médula ósea a partir de un sólo tipo de célula llamada **célula precursora hematopoyética pluripotencial (PHSC)** o **hemocitoblasto** (célula madre pluripotencial) definida como aquélla que puede dar origen a cualquier célula sanguínea y de mantener su propia existencia por divisiones (mitosis). A medida que se reproducen las **células madre pluripotenciales**, una pequeña parte de ellas permanece exactamente igual que la original en la médula ósea para mantener su aporte y el resto de las células reproducidas se diferencia hasta formar los otros tipos celulares maduros, proceso llamado **hematopoyesis**. Las células inmaduras se llaman **blastos**, algunos de ellos permanecen en la médula ósea para madurar y otros viajan a otras partes del cuerpo para madurar y ser funcionales. (*Cap. 10, p. 241-245 [3]*).

3.2.1. Glóbulos rojos y plaquetas

Los **glóbulos rojos** (eritrocitos) contienen hemoglobina que le da la característica tonalidad rojiza, tienen forma bicóncava, y están involucrados en el transporte de oxígeno y el metabolismo del hierro. (*Cap. 10, p. 235 [3]*). Los **trombocitos** (plaquetas) se forman por la diferenciación de **megacariocitos por la fragmentación de su citoplasma y cuyos fragmentos alcanzan los vasos sanguíneos desde la médula ósea**. (*Cap. 10, p. 250 [3]*). Estos dos grupos de células no están incluidos en el trabajo, ya que derivan en patologías diferentes a las que se estudian, como las anemias en los eritrocitos o problemas de coagulación en las plaquetas. Por lo tanto, se centra el trabajo en los **leucocitos** o **glóbulos blancos**.

3.2.2. Glóbulos blancos

Los **glóbulos blancos** se clasifican según su contenido granular en el citoplasma visible al microscopio óptico, así pues, existen los **granulares** y **no granulares.** (*Cap. 10, p. 233 [3]*).

El grupo de glóbulos blancos denominados **granulocitos** (***Figura 1***) se clasifica según las características de tinción de sus gránulos en:

- **A. Neutrófilos**: los más abundantes y los primeros en llegar a una zona infectada.
- **E. Basófilos:** en poca cantidad y presentes en respuestas inmunes.
- **C. Eosinófilos**: atacan parásitos e intervienen en reacciones alérgicas.

Los niveles normales de neutrófilos son de un 50-60%, entre 1-4% de eosinófilos y de menos del 2% de basófilos. (*Cap. 10, p. 26-2393 [3]*).

Figura 1. Eosinófilo, Neutrófilo y Basófilo. Fuente: http://leucocitos.org/

El grupo de los **no granulocitos** (***Figura 2***) no tiene gránulos en sus membranas celulares y se dividen en **linfocitos** y **monocitos:** (*Ver Cap. 10, p. 239-240 [3]*) y que constituyen asimismo las células mononucleadas.

- **B. Monocitos:** se producen en la medula ósea a partir de los monoblastos, circulan por el torrente sanguíneo algunos días para posteriormente desplazarse a tejidos de todo el cuerpo. Pueden segregar **antígenos** (sustancia que estimula una respuesta inmune) para que actúen los linfocitos. Algunos monocitos maduran a **macrófagos**, que son leucocitos especializados en digerir material extraño o defectuoso en los tejidos.
- **D. Linfocitos:** incluyen los **linfocitos T** y **B**, defienden al organismo de infecciones, ya que son capaces de detectar elementos extraños y las **células NK** (Natural Killer) que destruyen células infectadas.

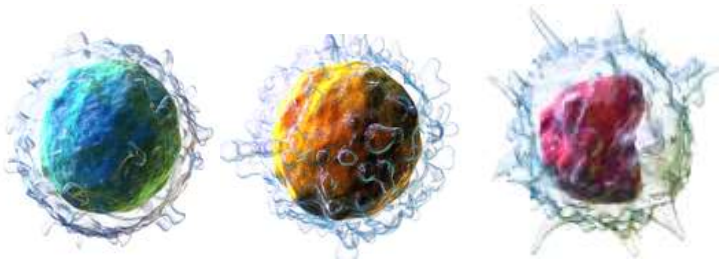


Figura 2. Linfocito B, Linfocito T, Monocito. Fuente: http://wikipedia.org/

3.2.3. Hematopoyesis

La **hematopoyesis** (***Figura 3***) es el proceso de maduración de los tres grupos celulares sanguíneos, la maduración de los eritrocitos se denomina **eritropoyesis**, la de las plaquetas, **trombopoyesis.** Pero como se comentó anteriormente, el estudio se centra en los glóbulos blancos.

Para el otro grupo celular, los glóbulos blancos, los procesos de maduración se distinguen en: **granulopoyesis** para los granulocitos, **monocitopoyesis** para los monocitos y **linfopoyesis** para los linfocitos. Las células madre hematopoyéticas pluripotenciales generan dos tipos de células madre de glóbulos blancos: **células madre linfoides multipotentes** o **linfoblastos** y **células madre mieloides multipotentes** o **mieloblastos**. Se les dice multipotentes a diferencia de sus precursoras que son pluripotentes porque sólo pueden diferenciar linfocitos, y el resto de células mieloides respectivamente, en cambio las pluripotentes pueden originar cualquier tipo celular. Así pues, en la diferenciación y maduración de glóbulos blancos se distinguen dos vías: la **linfoide** y la **mieloide**. (*Cap. 10, p. 246-251 [3]*).

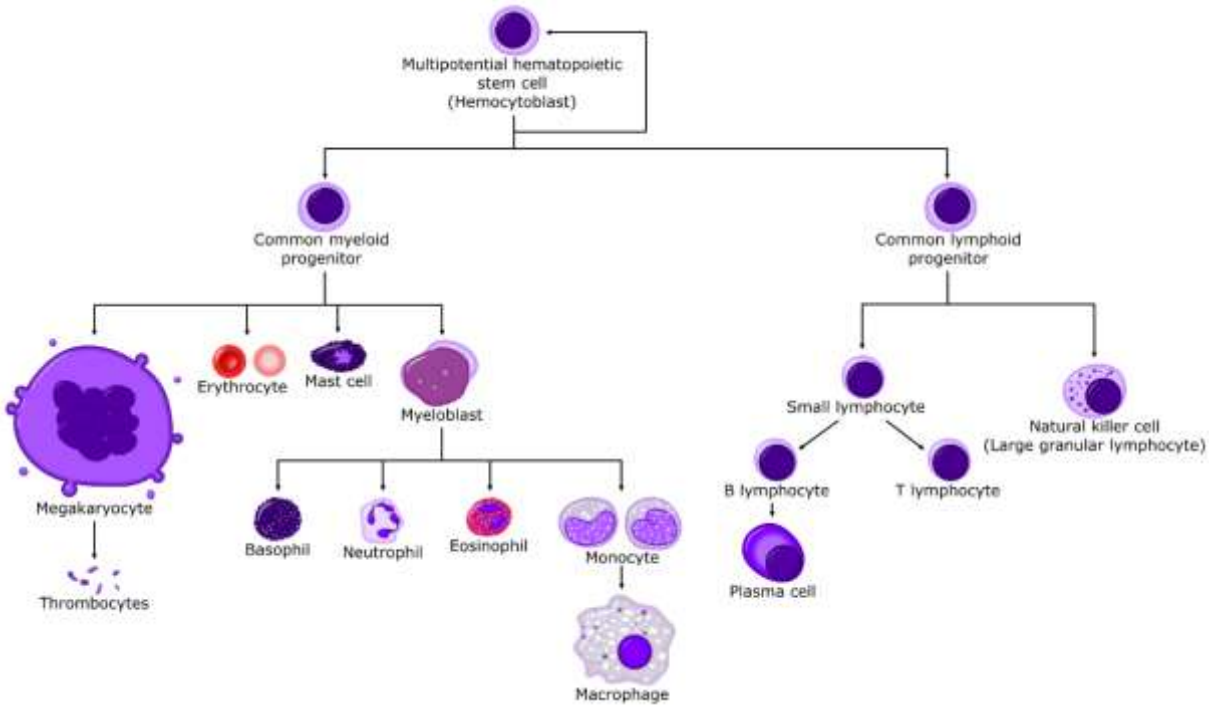


Figura 3. Hematopoyesis para la diferenciación y maduración de células sanguíneas. Fuente: https://classconnection.s3.amazonaws.com/10/flashcards/2422010/png/hematopoiesis_simple11360383179429.png

3.2.4. Vía linfoide

En la **vía linfoide** (**linfopoyesis**) las células madre hematopoyéticas multipotentes de la médula ósea, por una parte maduran a **células NK** (natural killer) y por otra, se diferencian en **células madre linfoides multipotentes** en **tejido u órganos linfáticos** (*Ver Apartado 3.3*), diferenciándose en:

- Células madre de linfocitos B: se diferencian en la médula ósea hasta madurar linfocitos **B**.
- Células madre de linfocitos T: se diferencian en el timo hasta madurar linfocitos **T**.

3.2.5. Vía mieloide

En la **vía mieloide** las células madre hematopoyéticas multipotentes diferencian **células madre mieloides multipotentes** que generan **Unidades Formadores de Colonias** (UFC) y que también son células madre pero más especializadas. Las UFC formadas son:

- **UFC-GM bipotente:** que diferencia a **UFC-G unipotente** que madura **granulocitos neutrófilos** (parte de la granulopoyesis) y **UFC-M unipotente** que madura a **monocitos** (monocitopoyesis) y que éstos pueden diferenciarse a **macrófagos**.
- **UFC-Bas:** que diferencian mieloblastos basófilos y maduran a basófilos (granulopoyesis).
- **UFC-Eo:** que diferencian mieloblastos eosinófilos y maduran a eosinófilos (granulopoyesis).
- **UFC-Meg**: que diferencian a megacarioblastos y maduran a plaquetas (trombopoyesis).
- **UFC-E**: que diferencian a proeritroblastos y maduran a eritrocitos (eritropoyesis)

3.3. Tejidos y Órganos linfoides: Sistema linfático

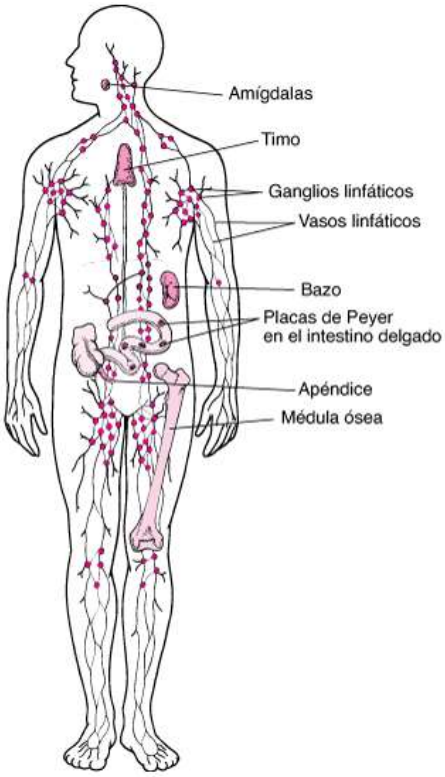


Figura 4. Sistema Linfático *[3]*

El **sistema linfático** forma parte del sistema inmunitario. Está compuesto por ganglios linfáticos e interconectado por los vasos linfáticos que contienen la linfa que nutre a los tejidos (***Figura 4***).

Los **tejidos** y **órganos linfoides primarios** están formados por la **médula ósea** y el **timo**, donde maduran las células madre de los linfocitos. Los linfocitos T se originan de las células madre nacidas en el timo, mientras que los linfocitos B nacen de sus células madre nacidas en la médula ósea. (*Cap. 16, p. 388 [3]*).

Los **tejidos** y **órganos linfoides secundarios** son las zonas dónde se producen las reacciones del **sistema inmunitario,** formados por los ganglios linfáticos, el bazo y agrupaciones de tejido linfoide asociado a las mucosas como las amígdalas, las placas de Peyer y el apéndice. (*Cap. 16, p. 388 [3]*).

3.4. Estudio de la sangre y frotis sanguíneo

Para el diagnóstico de muchas enfermedades es de gran importancia la observación de la composición y morfología celular de la sangre. En algunas casos como leucemias crónicas y agudas

puede llegar a confirmar el diagnóstico (*p.17 [2]*), para ello, se realizan **frotis sanguíneos**, que son extendidos de sangre que se preparan situando una gota de sangre en un portaobjetos, presionando con otro portaobjetos en ángulo, extendiendo la gota, y de nuevo, desplazando en sentido contrario, obteniendo así una capa fina de sangre. Tras el secado al aire, por calor o por algún reactivo, el extendido se fija y se tiñe por distintos métodos, normalmente se utiliza la tinción May-Grünwald-Giemsa, que contiene cosina y azul de metileno. (*Cap. 16, p. 234 [3]*).

Observando al microscopio un frotis normal, se observa un 99% de eritrocitos. La proporción de glóbulos blancos en personas sanas suele ser bastante constante con promedios de: 60% de neutrófilos, 3% de eosinófilos, 0,5% de basófilos, 5% de monocitos y 30% de linfocitos. (*Cap. 16, p. 234 [3]*).

En el torrente sanguíneo se pueden encontrar algunos leucocitos en apoptosis (muertos o moribundos) especialmente en frotis de pacientes con infecciones tales como la mononucleosis infecciosa u otras. En la ***Figura 5*** se muestran los cinco tipos de leucocitos teñidos mostrados al microscopio:

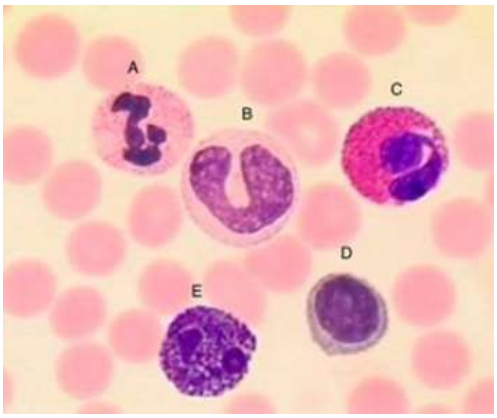


Figura 5. Vista al microscopio de células obtenidos mediante frotis sanguíneo de sangre periférica. Fuente: https://atlasudemhsito.blogspot.com.es/2012/04/sangre.html

3.5. Enfermedades de los glóbulos blancos

A continuación se detallan las enfermedades más importantes asociadas a cada tipo de glóbulo blanco, muchas de estas patologías no son visibles en frotis sanguíneo, ya que las células patógenas no están localizadas en el torrente sanguíneo. También es necesario realizar un recuento de cada tipo celular. Se debe tener en cuenta, que pueden existir glóbulos blancos anómalos de distinto tipo y en diferente proporción de forma simultánea.

3.5.1. Trastornos de los granulocitos

- **Neutrófilos**: las anomalías cualitativas y funcionales de los neutrófilos implican un amplio conjunto de enfermedades en las que encontramos la leucemia mieloide aguda y crónica, y anomalías funcionales graves como infecciones por *Staphylococcus aureus, Klebsiella aerobader, Escherichia coli* y otros microorganismos positivos a la catalasa (*Parte III Cap. 31, p. 163-167 [2]*).

- **Eosinófilos:** sus enfermedades más comunes asociadas y no graves son el asma, alergias e infecciones por parásitos, y en raras ocasiones enfermedades graves como algunos linfomas, leucemia eosfinófila crónica, leucemia mieloide crónica. (*Parte III Cap. 34, p. 191 [2]*).
- **Basófilos**: cuando existe un elevado número de basófilos, esto se asocia a diferentes afecciones de la piel, infecciones víricas, artritis reumatoide y diferentes patologías más graves como la leucemia mielogénica crónica, leucemia mieloide aguda, leucemia promielocítica aguda con basófilos maduros y leucemia basófila aguda, entre otras afectaciones. (*Parte III Cap. 35, p. 199 [2]*).

3.5.2. Trastornos de los linfocitos

La Organización Mundial de la Salud en el 2016 actualizó la clasificación de las patologías de la vía linfoide llamada REAL (Revised European-American Lymphoma Classification), que contiene una amplia gama de patologías clínicamente diferentes, algunas son:

- **Linfocitos T**: infecciones de cualquier tipo (bacterias, virus, hongos), inmunodeficiencias (poca capacidad de defensa ante agentes infecciosos), y éstas pueden estar asociadas o no a, linfomas de diferentes tipos e indoloros, linfomas cutáneos y linfomas agresivos tipo T. (*Parte VIII Cap. 54, p. 380-381 [2]*).
- **Linfocitos B**: al igual que el anterior, está asociado a linfomas indoloros y agresivos de tipo B de diferentes tipos, como el linfoma de Burkitt y el linfoma de Hodgkin. (*Parte VIII Cap. 54, p. 380-381 [2]*).

3.6. Metodología a aplicar

Una vez comprendidas las bases biológicas del estudio y comprendiendo la necesidad de estudiar la morfología de las células para los objetivos anteriormente especificados, se procede a enumerar la metodología utilizada:

- **FASE I (Capítulo 4): Análisis del conjunto de datos** a estudiar y preprocesado previo a la clasificación: esta fase es importantísima, ya que el conjunto de datos es el valor más preciado para unos resultados satisfactorios y sin desvirtuación de la información aportada. **Análisis de la técnica de aprendizaje automático más adecuada** para el conjunto de datos anterior.
- **FASE II (Capítulos 5, 6 y 7):** Creación de diferentes **algoritmos** con CNN y diferentes técnicas de menor a mayor complejidad. **Entrenamiento** de algoritmos y evaluación final de cada experimento para los diferentes modelos de redes neuronales convoluciones.
- **FASE III (Capítulo 9): Comparación** de resultados entre algoritmos para definir el más eficiente.

4. ANÁLISIS DEL CONJUNTO DE DATOS

4.1. Distribución y agrupación del conjunto de datos

Se dispone de un conjunto de datos que contiene dieciocho directorios que corresponden a diferentes tipos de patologías, dentro de cada una de estas carpetas, existen diferentes carpetas, que corresponden a los frotis realizados. Cada frotis contiene un número variable de imágenes, las cuáles están asociadas a linfocitos normales, anormales, reactivos y blastos, como el objetivo no es clasificar estas dieciocho clases, es necesario agrupar estos directorios en los cuatro grupos mencionados. Para ello se dispone también del archivo "**infos.csv**", que contiene la estructura de directorios anteriormente comentada. Todas las imágenes tienen un tamaño de 1200 x 900 pixeles.

La primera tarea es, distribuir las 18 carpetas y agruparlas en los cuatro grupos que queremos clasificar. Se sabe a qué tipo corresponde cada patología por la lectura del archivo "**infos.csv**" (***Tabla 1***).

Para la preparación de las imágenes se crea el algoritmo "**1_Data_Upload.ipynb**", que distribuirá todas las imágenes en sus grupos correspondientes, leyendo la estructura de directorios original del fichero "**infos.csv**" y creando una nueva estructura de directorios, cuya carpeta principal es **upload_dataset**, y como subdirectorios las carpetas: **VARIANT_LYMPHOCYTE** (reactivo), **BLAST** (blasto), **ATYPHICAL_LYMPHOCYTE** (linfocitos anormal) y **LYMPHOCYTE** (linfocitos normal).

Cada uno de estos directorios destino contendrán las imágenes de los frotis correspondientes a las patologías que afectan a ese tipo de célula. En total se disponen de 9.780 imágenes de 1200 x 900 píxeles.

En la ***tabla 1***, se puede apreciar que la mayoría de imágenes que se tienen, pertenecen al grupo **ATYPHICAL_LYMPHOCYTE,** ya que se han agrupado las subcategorías PROLYMPHOCYTE (prolinfocitos), ATYPICAL (incluye diferentes linfomas de tipo B y T), y CP (células plasmáticas).

CLASE	SUBCLASE	PATOLOGÍA	Nº IMÁGENES
LYMPHOCYTE		NORMAL	1018
VARIANT_LYMPHOCYTE		CLR	981
BLAST		LAM_MIELOIDE LAL-B_NO_BURKITT	956
ATYPICAL_LYMPHOCYTE	ATYPICAL (5485)	HCL_VARIANTE LZME LCM_BLASTICO LF LCM_INDOLENTE HCL_TIPICA SEZARY_TIPICO SEZARY_LUTZNER LLGG_T LLC_TIPICA	6825
	CP (554)	MM	
	PROLYMPHOCYTE (786)	LPB LPT LLC_TIPICA	
		Total	9780

Tabla 1. Agrupación de directorios del conjunto de datos en cuatro clases

4.2. Análisis del conjunto de datos

Antes de aplicar cualquier técnica de clasificación en el conjunto de datos, es esencial analizarlos con el fin de obtener resultados fiables de clasificación. El conjunto de datos disponible tiene diferentes inconvenientes, el primero es que las imágenes son demasiado grandes para introducirlas en una red neuronal, ya que aumentan el coste computacional. Para resolver este problema se adaptará el tamaño de las imágenes antes de la entrada a la red, definiendo el tamaño de entrada el estándar para el modelo de red aplicado. Esta técnica además permite que los algoritmos sean independientes del método de obtención de imágenes, ya que cualquier tamaño de imagen original, se convierte a un tamaño en concreto. *[9]*

Por otro lado, se encuentra que cada imagen de 1200 x 900 píxeles, contiene la célula a clasificar y su fondo de imagen, como vemos en la ***Figura 6 izquierda***. La proporción de píxeles del fondo de la imagen, es mucho mayor que la proporción de píxeles de la célula a clasificar. Para una CNN esto se interpreta como ruido, es decir, para facilitar la clasificación, se recortará la imagen a 900 x 900 píxeles para eliminar el exceso de fondo que no nos aporta información relevante (***Figura 6 derecha***). Un algoritmo detectará por segmentación y filtros dónde está la célula de interés, ya que tiene un color más oscuro que el resto por tinción en el frotis. Después, se recortará la imagen original a 900 x 900 píxeles, y como se comentaba anteriormente, este tamaño se reducirá antes de su entrada en la CNN.

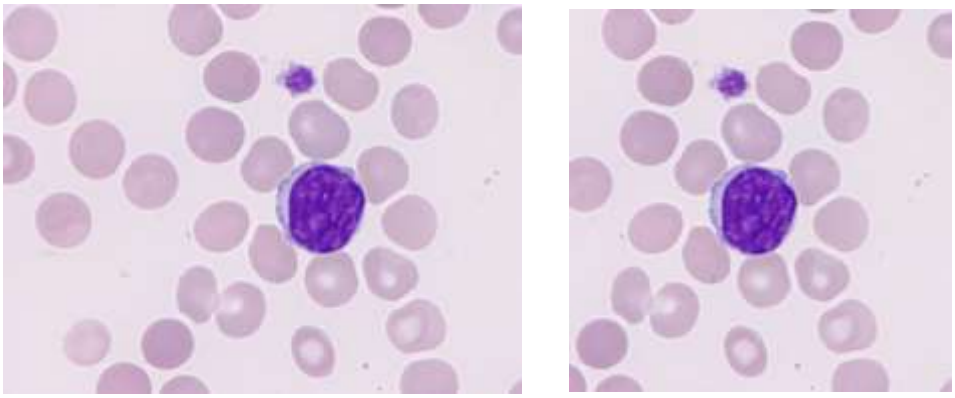


Figura 6. Imágenes tomadas con el objetivo X 100 de un frotis de sangre periférica teñido con MGG que muestra un linfocito en el centro. **Izquierda:** imagen original con exceso de fondo 1200x900 píxeles. **Derecha:** la misma imagen recortada y eliminando exceso de fondo 900x900 píxeles.

Otro problema a enfrentar, es que los datos **no están balanceados**, es decir, tenemos muchas imágenes de una clase y pocas de otras, lo que puede llevar a que el clasificador esté muy predispuesto a la clase mayoritaria. Para resolver este problema existen muchas técnicas: eliminar los datos de exceso de las clases mayoritarias, aumentar el número de datos de las clases minoritarias, realizar algoritmos insensibles al desequilibrio de clases y muchas otras. La elección del mejor método conllevaría evaluar todas las técnicas, lo que está fuera del alcance de este trabajo.

La opción adoptada es una **híbrida** que combina la reducción de imágenes de la clase mayoritaria, y aumento de datos de las clases minoritarias con el fin de conseguir un número igual de ejemplos de cada clase. Para ello se realiza un algoritmo que utiliza el mayor número posible de imágenes originales y evitar al máximo el aumento de datos, ya que éste, no proporciona más información a la red, sino que utiliza la ya existente y la transforma.

Antes de **balancear** los datos, se deben definir los conjuntos de **entrenamiento**, **validación** y **test** (***Figura 7***). Mediante algoritmo se divide el conjunto de datos en tres directorios: conjunto de entrenamiento (train) que servirá para entrenar la red neuronal, conjunto de validación (validation) utilizado para el ajuste de hiperparámetros y el conjunto de test (test) para la evaluación de los modelos. *[5]*

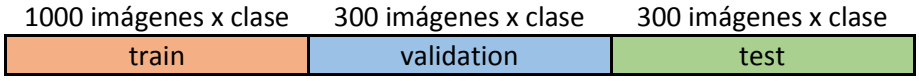


Figura 7. Distribución del conjunto de datos.

En CNN un problema muy importante que aparecerá desde el principio, es el **sobreajuste** de datos, es decir, tendremos demasiadas dimensiones de entrada en la red (características), y no todas estas características son relevantes para determinar la clasificación de la célula. Esto es debido, a que, para una CNN se necesita un conjunto de datos con cientos de miles de imágenes. En entornos experimentales y sanitarios, un conjunto de datos tan elevado, no es habitual, por lo que se requieren de otras técnicas que permitan solucionar, al menos en parte, dicho problema. Este inconveniente, se puede resolver o mejorar aumentando el conjunto de datos. Así pues, una de las técnicas que se aplicarán, es el **aumento de datos** en tiempo de ejecución en el conjunto de datos balanceado. *[4]*

Por lo tanto, la resolución del problema de clasificación del conjunto de datos dado, a cuatro clases de células, **no es un problema trivial**, pero las técnicas avanzadas en análisis profundo de redes neuronales convolucionales (**Deep Learning**), nos permitirán resolver este problema típico en Visión por Computador de forma aceptable, resolviendo los problemas que se detallan en la ***tabla 2.***

PROBLEMA	SOLUCIÓN
Imágenes de gran tamaño para CNN	Reducción del tamaño antes de la entrada a la CNN
Imágenes con exceso de fondo	Recorte de imágenes a 900 x 900 píxeles
Datos no balanceados	Método híbrido de balanceo de datos
Escasez de datos - Sobreajuste	Aumento de datos y regularización

Tabla 2. Problemas en el conjunto de datos a resolver y soluciones

La preparación de los datos (*Apartado 4.1*) y la resolución de los anteriores problemas se resuelven, ejecutando en orden los siguientes algoritmos:

1. **Upload_dataset.ipynb** → **upload_dataset**
2. **Split_dataset.ipynb** → **split_dataset**
3. **Cropping_dataset.ipynb** → **cropped_dataset**
4. **Balanced_dataset ipynb** → **balanced_dataset (conjunto de datos a clasificar)**

Los algoritmos se realizan por separado para tener mayor control y poder grabar en disco cada proceso y reducir el coste computacional. Una vez, comprobado el funcionamiento de los algoritmos, puede unirse en uno sólo y optimizarlo, para realizar sólo una grabación en disco.

Después de preparar el conjunto de datos, tendremos en disco el conjunto de datos balanceado "**balanced_dataset**", dividido en tres carpetas (train, validation y test), con 4000 imágenes para entrenamiento (train), 1200 imágenes para validación (validation) y 1200 imágenes para test (test), cada una de ellas con una subcarpeta por clase (***Tabla 3***). A partir de aquí podemos decidir si aplicar o no técnicas de aumento de datos en tiempo real, si se disponen de recursos técnicos que no hagan prohibitivo el coste computacional. Como las primeras pruebas se realizan en computadora personal, este conjunto de datos permitirá tener una entrada de datos balanceada en disco sin necesidad de aumentarlos, y para las pruebas definitivas en GPU utilizar el aumento de datos y comparar resultados.

balanced_dataset		TRAIN	VALIDATION	TEST
ATYPHICAL	ATYPHICAL	1000	300	300
BLAST	BLAST	1000	300	300
LYMPHOCYTE	LYMPHOCYTE	1000	300	300
VARIANT	VARIANT	1000	300	300
		4000	1200	1200

Tabla 3. Distribución del número de imágenes en grupos para entrenamiento, validación y test

A partir de aquí, ya se tiene disponible el conjunto de datos, para su clasificación y puede evaluarse qué técnicas implementar para la creación de algoritmos.

4.3. Análisis de la técnica adecuada a aplicar

El problema de clasificación de imágenes, parte de la tarea de asignar una etiqueta de un conjunto de categorías a cada imagen de entrada, por ejemplo, si tenemos imágenes de gatos y perros para clasificar, a cada imagen de entrada le correspondería la etiqueta "gato" o "perro". En un principio, parece una tarea trivial para un cerebro humano, pero debemos darle a un algoritmo la capacidad de interpretación de un **concepto**, es decir, aunque los gatos y los perros tienen: boca, ojos, orejas, patas, etc., hay características inequívocas que visualmente el cerebro rápidamente interpreta como "gato" o como "perro", esas características, que no las conocemos para el problema de clasificación de células, son las que nos interesan obtener para clasificar las imágenes correctamente.

A continuación analizamos las técnicas disponibles para el aprendizaje automático y justificar la técnica que se ha implementado:

- **Machine Learning:** basado en distancias, redes neuronales artificiales normales (ANN).
- **Deep Learning:** redes neuronales convolucionales (CNN).

Las imágenes digitales disponibles, están formadas por tres matrices, una para cada canal de color R, G, B (Red, Green, Blue), con ancho y alto de matriz, correspondiente al número de píxeles de ancho y alto de imagen. Lo que significa que para cada imagen tenemos 900 x 900 x 3 = 2.430.000 parámetros de entrada en la red neuronal. Si intentamos clasificar las imágenes con técnicas de **Machine Learning:** clasificadores basados en distancias o una red neuronal artificial normal (ANN), nos encontraríamos con los siguientes problemas *[8]* (***Figura 8***):

- **Variación del punto de vista**: una imagen se puede orientar de muchas maneras con respecto a la cámara.
- **Variación de escala:** las imágenes pueden estar en tamaños diferentes.
- **Deformación:** hay objetos que no son rígidos y pueden estar deformados.
- **Oclusión:** la zona de la imagen que se quiere clasificar puede estar oculta parcialmente.
- **Condiciones de iluminación:** las diferencias de iluminación resultan en valores de píxel muy diferentes aunque pertenezcan al mismo objeto.
- **Desorden de fondo:** las zonas de interés pueden mezclarse con el fondo de la imagen, lo que los hace difíciles de identificar.
- **Variación intercalase**: las zonas a clasificar puede pertenecer a una clase que contengan diferentes subclases.

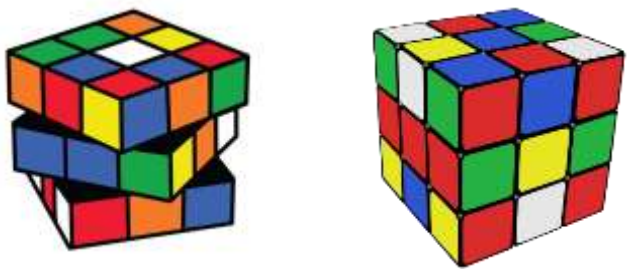


Figura 8: El mismo objeto está deformado y con diferente punto de vista.
Fuente: ztfnews.wordpress.com

Utilizar diferencia de imágenes (distancias), k-means, distancia euclidiana, vecino más cercano, etc., hace que una imagen de un gato en fondo azul y una rana en fondo azul, las clasifique igual, en lugar de clasificar “gato” y “rana”. Ya que la diferencia entre los píxeles del fondo será baja (Valor de 0 indica que las imágenes son iguales). Por lo tanto, esta técnica no es útil para la clasificación de imágenes ya que no clasifica de forma semántica o interpretativa, la realiza por distribuciones de color. En la imagen (***Figura 9***), se puede ver como imágenes de diferentes clases, quedan agrupadas por el mismo color de fondo, aunque las imágenes correspondan a clases diferentes.

Figura 9. Efecto del fondo en lugar de las diferencias de clase semántica. Fuente: https://cs231n.github.io/

A continuación se detallan los **problemas de clasificación de imágenes con Machine Learning** *[7]*:

1. **Coste computacional**, entrenamiento es rápido pero la evaluación de una nueva imagen es muy lenta, ya que debe comparar la imagen nueva con todas las imágenes de entrenamiento. Las redes neuronales profundas son costosas de entrenar pero una vez entrenadas es muy rápido clasificar un nuevo conjunto de test.
2. Con **Multidimensionalidad** de imágenes no se puede parametrizar la entrada con modelos lineales o linealmente separables, ya que el modelo es mucho más complejo. Aunque existen técnicas que sí pueden añadir no linealidades, no permiten solucionar el resto de problemas, ni modelar el modelo con la complejidad que se necesita.
3. Las imágenes **no son datos estructurados**, los estructurados como puede ser una base de datos relacional, o conjuntos de datos con estructura (por ejemplo el precio de una casa según el número de habitaciones y el tamaño), sí pueden clasificarse bien con Machine Learning, pero una imagen, no contiene información estructurada, ya que el valor de un píxel, no tiene por qué estar relacionado con los píxeles del entorno, ni de los píxeles de los otros canales.
4. Con ANN grandes, el entrenamiento sería extremadamente **lento**.

5. Un modelo con millones de parámetros correría un grave riesgo de **sobreajuste** del conjunto de entrenamiento.

Por otro lado si una red neuronal artificial normal, encuentra una oreja de un gato en la parte superior izquierda de una imagen a clasificar, el algoritmo deberá volver a aprender este concepto para cualquier otra parte de la imagen. Si una ANN es capaz de reconocer gatos con las patas en el suelo, no será capaz de reconocer gatos que tengan sus patas hacia arriba, deberá volver a aprender cada posición diferente que pueda tener un gato. Para poder aplicar ANN al problema de 2,43 millones de parámetros por cada imagen, teniendo 4000 imágenes de entrada, se necesitaría una ANN con muchísimas capas, lo que haría el coste computacional inviable, y tampoco resolvería los problemas anteriores.

Un correcto clasificador de imágenes debe ser insensible a estos problemas e interpretar el concepto visual que realiza el cerebro humano de forma generalizada para cualquier imagen de entrada y con un coste computacional aceptable. La técnica que permite resolver los problemas anteriores (en parte) son **las Redes Neuronales Convolucionales (CNN).** Para comprenderlas, a continuación se repasan los conceptos más importantes de las redes neuronales**.**

4.3.1. Neuronas Artificiales

Las unidades fundamentales de las redes neuronales son las **neuronas** (***Figura 10***), basadas en las neuronas biológicas, que multiplican los parámetros de entrada $\mathbf{x_i}$ por sus pesos $\mathbf{w_i}$ y realiza una suma ponderada de las entradas más el sesgo o error. Después aplica un clasificador binario no lineal o función de activación.

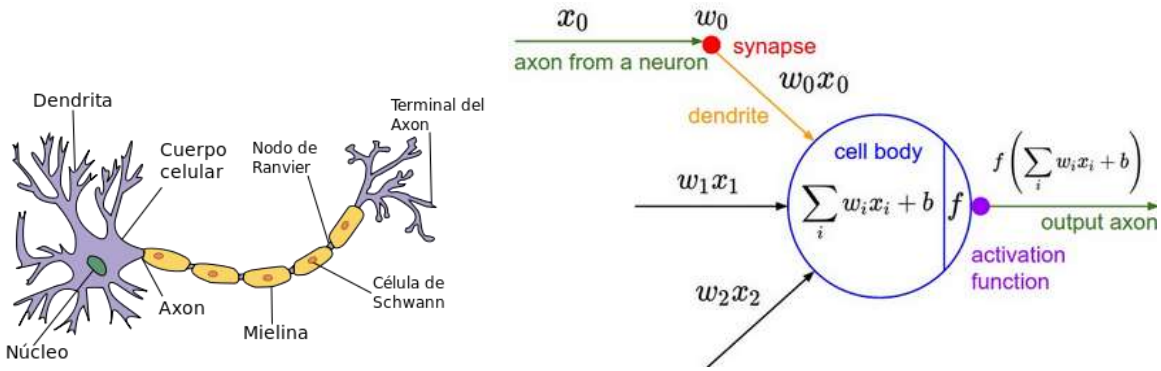


Figura 10. Neurona biológica y Neurona artificial: unidad básica de una red neuronal artificial. Fuente: http://www.wikipedia.org

4.3.2. Clasificador binario no lineal o Función de Activación

Las funciones de activación, permiten clasificar la entrada de números reales en un rango definido. Existen muchos clasificadores no lineales, algunos son:

- **Sigmoide Logística:** puede saturar la salida y anular gradientes en la retropropagación (derivada parcial=0), y si los pesos son muy grandes puede saturar las neuronas, lo que

provoca que el aprendizaje sea malo. Además no está centrada en 0, lo que puede producir un zigzag en el aprendizaje. La función **Softmax**, es una variante de la sigmoide logística, pero en lugar de ser binaria, permite clasificar problemas multiclase (***Figura 11***).

$$f(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

- **Sigmoide Hiperbólica:** da mejores resultados que la anterior porque está centrada en 0, aunque sigue mostrando los problemas de saturación y anulación de neuronas. (***Figura 11***).

$$f(x) = \tanh(x) \qquad (2)$$

Figura 11. Funciones de Activación. Izquierda: la función no lineal sigmoidea transforma la entrada para que oscilen entre [0,1] **Derecha:** la no linealidad del tanh transforma la entrada para que oscilen entre [-1,1].

- **ReLU** (Rectified Linear Unit): es muy útil en redes profundas porque acelera el entrenamiento en un factor de 6 comparado con las dos anteriores *[11]* y evita el problema de desvanecimiento de gradiente o anulación del gradiente (***Figura 12***).

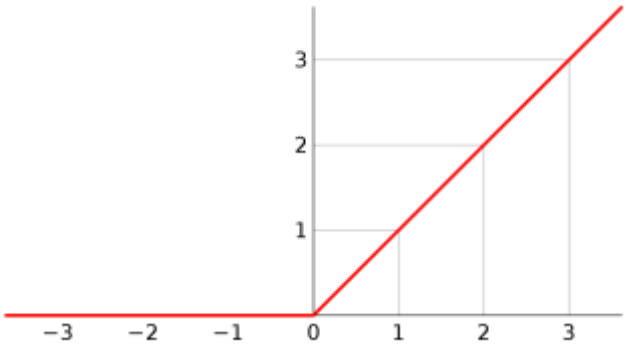


Figura 12. **Función de activación ReLU.** Es cero cuando x <0 y luego lineal con pendiente 1 cuando x> 0.

4.3.3. Red Neuronal Artificial

Una red neuronal artificial es un conjunto de neuronas artificiales que se conectan de forma no cíclica, las salidas de unas neuronas pueden ser la entrada de otras. Para redes neuronales normales, el tipo de capa más común es la capa completamente conectada (***Figura 13***) en la que las

neuronas entre dos capas adyacentes están conectadas por pares por completo, pero las neuronas dentro de una sola capa no tienen conexiones. Una red neuronal se estructura de la siguiente forma:

- **Capa de Entrada (input):** parámetros de entrada, asignación de pesos y funciones de activación
- **Capa/s Ocultas (hidden):** parámetros de entrada de la capa anterior, asignación de pesos y funciones de activación
- **Capa de Salida (output):** parámetros de salida son el resultado de la entrada de la capa anterior y la función de pérdida (*Apartado 7.3.4*), que busca los valores de los pesos $\mathbf{w}_i$ que minimizan el error.

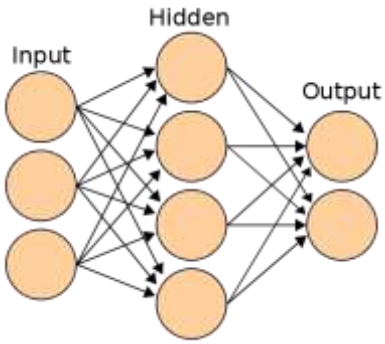


Figura 13: Red neuronal completamente conectada.

La red tiene 4 + 2 = 6 neuronas (sin contar las entradas), [3 x 4] + [4 x 2] = 20 ponderaciones y 4 + 2 = 6 sesgos, para un total de 26 parámetros aprendibles.

Si se sabe que tenemos 2,4 millones de parámetros por imagen (sin reducirla), y queremos diseñar una red neuronal, se deberá tener en cuenta este dato para diseñar su tamaño.

4.3.4. Red Neuronal Convolucional

Las redes convolucionales actuales contienen del orden de 100 millones de parámetros y normalmente están formadas por aproximadamente 10-20 capas (de ahí el aprendizaje profundo o **Deep Learning** *[5]*. Las CNN son muy parecidas a las ANN, con la diferencia que se supone explícitamente que **los datos de entrada son imágenes**. Aprovechan esta particularidad y restringen la arquitectura de una manera más sensata.

Las capas de una CNN tienen neuronas dispuestas en 3 dimensiones: ancho, alto, profundidad. Las neuronas en una capa solo se conectarán a una pequeña región de la capa anterior, en lugar de todas las neuronas de una manera totalmente conectada como ocurre en las ANN. Además, en la capa de salida final de la arquitectura CNN reduce la imagen completa a un solo vector de pesos de clase, dispuestos a lo largo de la dimensión de profundidad (***Figura 14***).

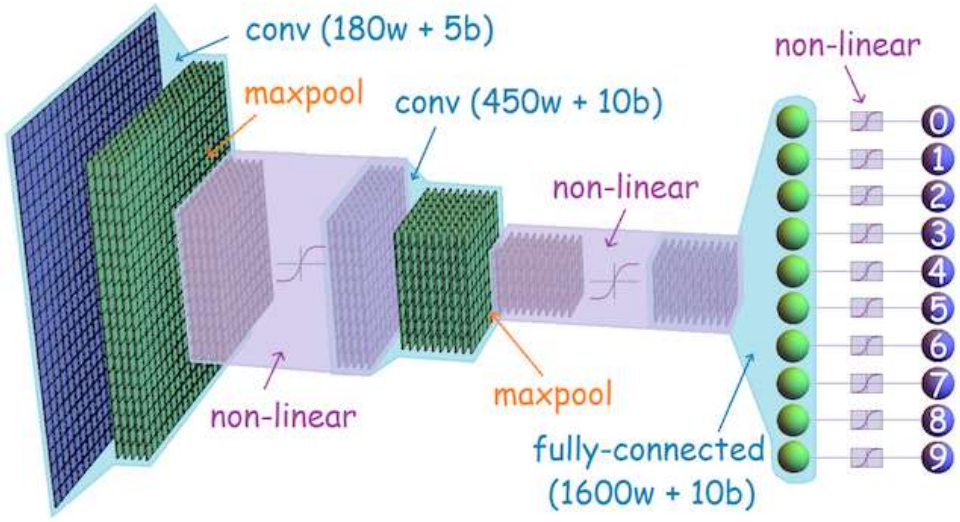


Figura 14. Transformación de la imagen de entrada. Una CNN transforma el tensor que representa la imagen de entrada con sus tres canales de color a medida que atraviesa las capas. Fuente: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

Capa Convolucional

Las capas convolucionales son los componentes básicos de las CNN y dónde se realiza el mayor coste computacional para el cálculo de pesos. Los parámetros de las capas convolucionales consisten en un conjunto de filtros que se pueden aprender. Cada filtro es una ventana de unos pocos píxeles de alto y ancho, y la entrada de la capa convolucional sería de *ancho_filtro* x *alto_filtro* x *canales*. Por ejemplo, un filtro como parámetro de entrada podría ser de 5x5x3, 5 píxeles de ancho, 5 de alto y 3 píxeles uno para cada canal (RGB). En la convolución, el filtro se desliza por toda la imagen calculando el valor máximo que coincida con el filtro, produciendo un mapa de activación bidimensional (***Figura 15***), que proporciona la respuesta al filtro en cada posición espacial. La red aprende cuando los filtros que se activan ven algún tipo de característica visual, como un borde de alguna orientación o una mancha de algún color en la capa. Para todas las capas convolucionales se apilarán estos mapas de activación a lo largo de la dimensión de profundidad y se producirá el volumen de salida. *[11]*

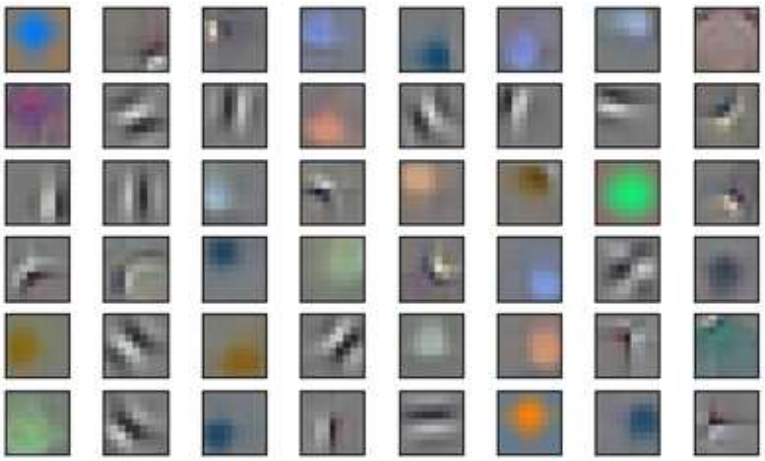


Figura 15. Ejemplo de filtros aprendidos en una CNN. Fuente cs.toronto.edu

Cuando se trata con entradas de muchas dimensiones como en imágenes, no es práctico conectar las neuronas a todas las neuronas de la capa anterior. Es mejor conectar cada neurona a solo una región local del volumen de entrada. Una capa densamente conectada como en las redes neuronales normales **ANN**, aprenden **patrones globales** en su espacio de características de entrada (todos los píxeles), en cambio, las capas de convolución de las **CNN** (localmente conectadas), aprenden **patrones locales** (***Figura 16***). La extensión espacial de esta conectividad es un hiperparámetro llamado **campo receptivo de la neurona** (el tamaño del filtro). El tamaño del filtro a lo largo del eje de profundidad siempre es igual a la profundidad del volumen de entrada. *[5]*

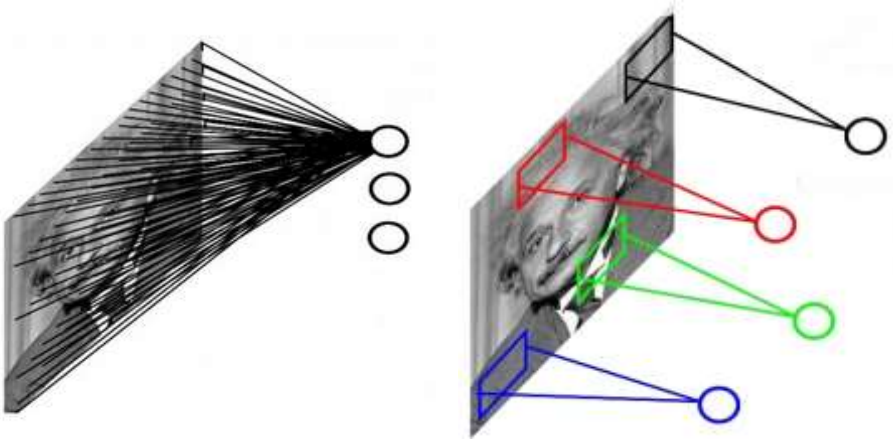


Figura 16. Comparación de dos redes neuronales. Izquierda: completamente conectada**. Derecha:** localmente conectada. Fuente: https://legacy.gitbook.com

Las capas convolucionales permiten que un filtro aprendido de forma local en una parte de la imagen, automáticamente quede aprendido para el resto de zonas de la imagen, además puede aprender jerarquías de patrones (***Figura 17***), siendo los patrones cerca de la entrada más generales, y los cercanos a la salida, más especializados a las imágenes a clasificar.

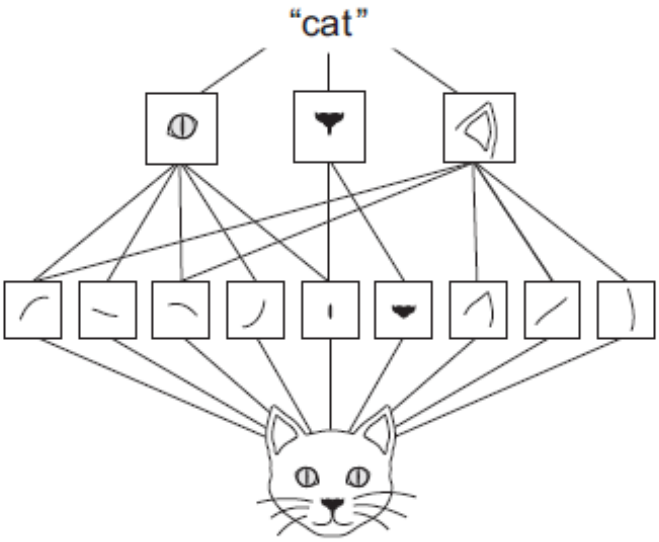


Figura 17. Jerarquía de filtros aprendidos *[6]*

Capa Pool

Las capas **Pool** (agrupación) tienen como objetivo submuestrear, es decir, reducir la imagen de entrada para disminuir la carga computacional, el uso de la memoria y el número de parámetros (lo que limita el riesgo de sobreajuste). La reducción del tamaño de la imagen de entrada también hace que la red neuronal tolere un poco de cambio de imagen (invariancia de ubicación). *[10]*

Al igual que en las capas convolucionales, cada neurona en una capa **Pool,** se conecta a las salidas de un número limitado de neuronas en la capa anterior, ubicada dentro de un pequeño campo receptivo rectangular. Pero, una neurona de capa **Pool** no tiene pesos, todo lo que hace es agregar las entradas usando una función de agregación como **máximo** o **promedio**. Las capas **MaxPooling2D** aplicadas en los experimentos de este trabajo (capas de agrupamiento máximo), extraen ventanas de las características de entrada y generan el valor máximo de cada canal. Esto hace que las características de entrada se reduzcan a la mitad, lo que permite reducir el número de coeficientes a procesar. *[5]*

Capa de Regularización

Existen variados métodos de regularización que sirven para reducir el sobreajuste del modelo a los datos durante el entrenamiento. Los experimentos realizados, o no utilizan regularización o utilizan la regularización **Dropout** (apagado aleatorio) que es simple y altamente efectivo *[12]*. En el entrenamiento, algunas neuronas se mantienen activas con una probabilidad ***p*** y las otras son apagadas a 0. Esto permite que sólo se actualicen los pesos de las neuronas activas, acelerando el aprendizaje y reduciendo el sobreajuste.

Clasificador

Un clasificador simple asigna una etiqueta de un conjunto de categorías a una muestra. Existen cuatro tipos de clasificadores, dependiendo de la naturaleza de los datos a clasificar:

- **Clasificador binario:** clasificación de dos clases, es decir, asigna una etiqueta a una muestra de dos posibles etiquetas asignables. Por ejemplo: asignar a una imagen etiqueta “gato” o “perro”.
- **Clasificador multiclase:** es como el anterior pero permite clasificar más de dos clases. Este trabajo se centra en este tipo de clasificador, ya que queremos clasificar cuatro tipos de células.
- **Clasificador multietiqueta:** permite asignar varias etiquetas a la misma clase. Por ejemplo, un clasificador que pueda reconocer “perros”, “gatos” y “loros”, en una imagen que contenga “perros” y “loros” asignaría las etiquetas [1, 0, 1]. Valor 1 para los que sí están en la imagen: “perros” y “loros” y 0 para “gatos”.
- **Clasificador de múltiples salidas:** es un clasificador parecido al anterior, pero cada etiqueta puede ser multiclase.

Compilación de los modelos antes del entrenamiento

En los experimentos, se hará referencia a varios conceptos a la hora de entrenar los modelos, que aquí se enumeran para describir su función:

- **Función de Pérdida** (los function): define cómo la red podrá medir su rendimiento en los datos de entrenamiento y como podrá orientarse en la dirección correcta. Como se comentó anteriormente, el tipo de clasificador a utilizar será **multiclase**, ya que queremos clasificar cuatro tipos de células, según (pag.114 Tabla 4.1 *[5]*), podemos utilizar las funciones de pérdida: *categorical_crossentropy* y *binary_crossentropy*, dependiendo de si asignamos una o varias etiquetas por muestra.

- **Optimizador** (optimizer): mecanismo que permite que la red se actualice a sí misma en función de los datos y la función de pérdida. Permite optimizar la salida, haciendo que la contribución de pesos de cada neurona que genera menor error se actualice.

- **Métricas** (metrics): permite evaluar la red durante el entrenamiento. En los experimentos se utilizará el valor de pérdida *loss* y la exactitud *accuracy* (imágenes clasificadas correctamente en tanto por ciento).

La ***figura 18*** muestra un esquema general del procedimiento de aprendizaje y evaluación en las CNN.

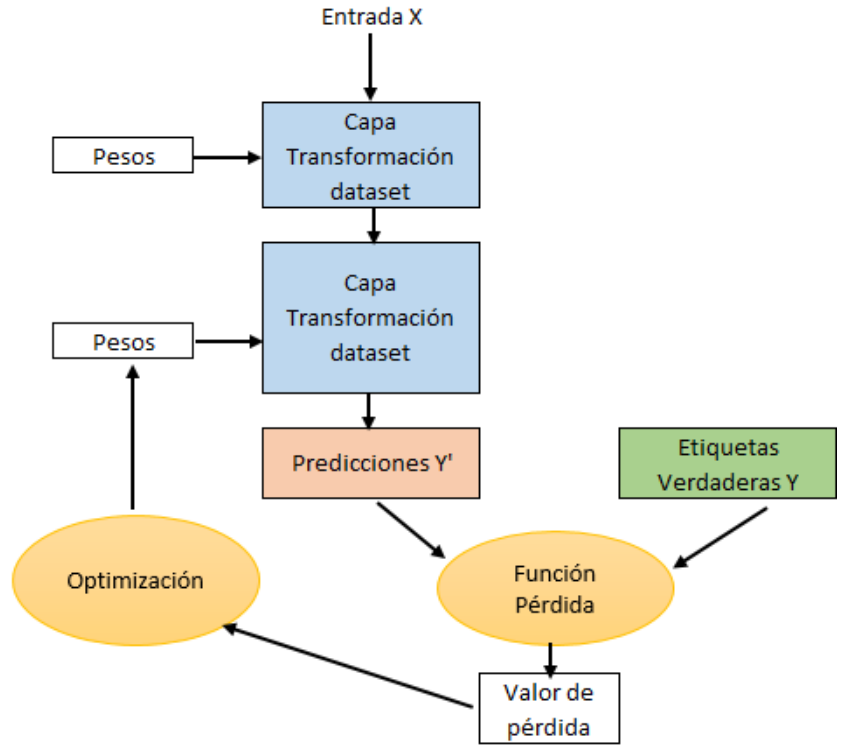


Figura 18. Procesos en el aprendizaje de una CNN.

Se ha visto una visión general de las Redes Neuronales Artificiales, las Redes Neuronales Convolucionales y los conceptos básicos que se aplicarán en los experimentos. Una vez que se ha especificado la justificación del uso de redes neuronales convolucionales (CNN) para la resolución del problema de clasificación de imágenes que se quiere abordar y su funcionamiento básico, se procede a detallar los algoritmos realizados con CNN **(FASE II)**.

5. CREACIÓN DE RED NEURONAL CONVOLUCIONAL DESDE CERO

La primera técnica implementada es la creación de un algoritmo con una red neuronal convolucional entrenada desde cero, y que, permitirá tener una aproximación de la eficiencia que se debería superar con algoritmos más complejos. Los pasos a aplicar en los algoritmos son los siguientes:

1. Carga del conjunto de datos, haciendo referencia a los directorios: train, validation y test
2. Creación de la red neuronal convolucional CNN para clasificar 4 clases
3. Preprocesado de imágenes de entrada con y sin aumento de datos
4. Compilar el modelo
5. Entrenar el modelo con los datos de entrenamiento (train) y validarlo con los datos de validación (validation)
6. Mostrar gráficas de efectividad (*Accuracy*) y pérdida (*Loss*)
7. Evaluar el modelo con los datos de test

El código implementado en el experimento 1 y 2, son adaptaciones de (*Cap.5.2 [5]*).

5.1. Experimento 1: CNN sin aumento de datos

La red neuronal creada recibe imágenes de entrada de 224 x 224 píxeles. Es un modelo secuencial, que consta de tres bloques, los dos primeros bloques tienen la siguiente estructura: dos capas convolucionales seguidas de una capa *MaxPool* y otra capa *Dropout*, que intenta evitar el sobreajuste. El último bloque, aplana la salida, conecta con una capa densa, su función de activación **ReLu** (*Apartado 7.3.1*), aplica la regularización **Dropout** (*Apartado 7.3.2*) y termina con el clasificador **Softmax** que es como la clasificación *Sigmoidal Logística* pero multiclase (*Apartado 7.3.1*).

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten


def createModel(nClasses, height, width, channels):
    model = Sequential()
    #Bloque 1
    model.add(Conv2D(32,(3,3),padding='same',activation='relu',
       input_shape=(height,width,channels)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    #Bloque 2
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    #Bloque 3
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    #Bloque 4
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(nClasses, activation='softmax'))

    return model
```

Si se tienen imágenes de entrada de 224 x 224 píxeles y tres canales de color, en la CNN debemos introducir tensores (vectores) de ese tamaño, a medida que avanzan los datos en la CNN se van transformando, perdiendo tamaño (alto x ancho) y ganando en profundidad. A continuación, se muestra un ejemplo de cómo se transforman los datos en el primer bloque convolucional.

Siendo: N = tamaño de entrada F = nº de filtros

- **Capa de entrada**: aplica 32 filtros de 3x3 a capa convolucional con función de activación "ReLu"a las imágenes de entrada. 229 x 229 x 32 N1 = 229

  N x N x F = Parámetros de entrada en capa inicial (3)

- **Capa Convolucional:** aplica 32 filtros de 3x3 a capa convolucional con función de activación. (229-3 +1) x (229-3)+1 x 32 = 227 x 227 x 32. N2 = 227

  (N1-F)+1 x (N1-F) +1 x F = tamaño de salida en capa convolucional (4)

- **Capa de Agrupamiento máximo**: Aplica MaxPooling2D. Divide el anterior entre dos. 227/2 x 227/2 x 32 = 113 x 113 x 32 N3 = 113

  N2/2 x N2/2 x F = tamaño de salida en capa MaxPooling2D (5)

A lo largo de la evolución de los datos a través de las capas se van transformando. Las capas "**dropout"**, no modifican el tamaño de los datos, pero se añaden para evitar (en parte) el sobreajuste, por otro lado, las capa "**flatten**" aplanan el vector anterior en forma de cubo (tensor), a un vector de una dimensión, por ejemplo, la capa "**dropout_3**" de tamaño 26x26x64, después de la siguiente capa "**flatten**" se convierte en un vector de 26x26x64=43.264 parámetros. Las capas "**dense**" conectan ese vector con las neuronas de la capa anterior, y resulta en 43.264x512=22.151.680 parámetros. Esto, nos indica, que la CNN creada, puede entrenar y aprender más **de 22 millones de parámetros**.

Después del preprocesado sin aumento de datos, compilamos el modelo con el optimizador "RMSProp".

```
from keras import optimizers
epochs = 30
model.compile(
    optimizer = 'rmsprop',
    loss = 'categorical_crossentropy',
    metrics = ['accuracy'])
```

En otro de los experimentos con la misma red, compilamos con el optimizador "SGD", con el fin de compararlos.

```
from keras import optimizers
epochs = 30
model.compile(
    optimizer = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True),
    loss ='categorical_crossentropy',
    metrics = ['acc'])
```

A continuación, entrenamos el modelo validándolo con el conjunto de datos de validación.

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch = nb_train//batch_size,
    epochs = epochs,
    verbose = 1,
    validation_steps = nb_validation//batch_size,
    validation_data = validation_generator )
```

Como resultado obtenemos:

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	Acc	Loss	Acc	Loss	Acc	Sobreajuste	Regulariz	Epoch	Optimizer
0,17	95,01	1,24	63,02	1,22	63,50	a 10 epochs	Dropout	30	RMSProp
12,13	24,77	12,14	24,65	12,09	25,00	Si	Dropout	30	SGD

Tabla 4. **Experimento 1**. CNN desde cero sin aumento de datos

Podemos apreciar que a algoritmos iguales, y modificando el optimizador, obtenemos resultados muy diferentes, esto es debido, a que el optimizador **SDG** (Stochastic Gradient Descent) no es útil en este caso *(p. 117 [4])*. Por lo tanto, vemos que no es apto para este experimento.

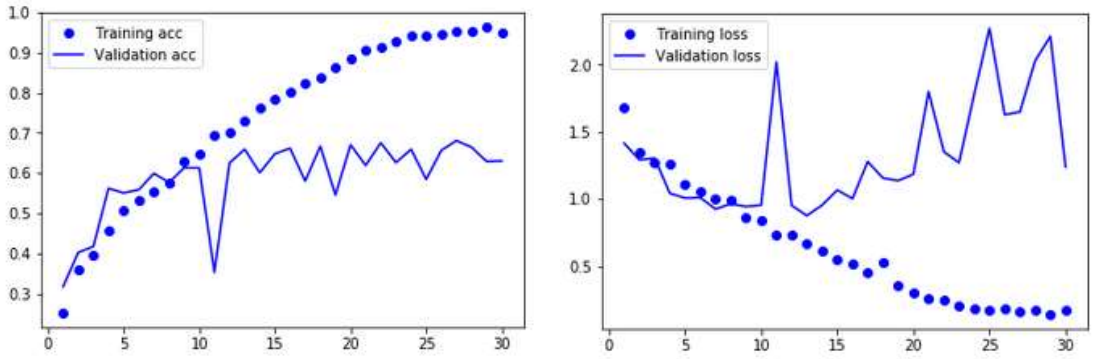


Figura 19. Experimento 1. Gráfica Exactitud y pérdida. Con optimizador "RMSProp"

A partir de aquí se sabe, que se debería superar una exactitud del **63,50 %** y que se debe mejorar el sobreajuste que aparece aproximadamente a las 10 épocas de entrenamiento

5.2. Experimento 2: CNN con aumento de datos

En este experimento se aplica el algoritmo anterior añadiendo aumento de datos en el preprocesado. Permite que cuando generamos grupos de imágenes aleatorias, se modifiquen las imágenes de entrada, creando más imágenes en tiempo de ejecución. Estas imágenes son modificaciones de las imágenes leídas: como traslaciones, giros, transposiciones, cambios de escala, etc. Y que permiten paliar (en parte) el sobreajuste. Sólo se realiza aumento de datos en el conjunto de entrenamiento, nunca en los de validación y test.

```
#DATA AUGMENTATION
#Transformador de imágenes de forma aleatoria SOLO A TRAIN

from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale = 1./255,              #Normalizamos el valor de pixel
    rotation_range = 40,           #rotamos imagne 40 grados
    width_shift_range =0.2,        #aplica una fracción del ancho de la imagen orig
    height_shift_range =0.2,       #aplica una fracción del alto de la imagen orig
    shear_range = 0.2,             #aplica al azar operaciones de corte
    zoom_range = 0.2,              #aplica zoom aleatorio dentro de las imagenes
    horizontal_flip = True,        #invierte aleatoriamente la mitad de las imágenes
    fill_mode = 'nearest')
```

Después de compilar con los dos optimizadores anteriores y entrenar el modelo, obtenemos:

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Sobreajuste**	**Regulariz**	**Epoch**	**Optimizer**
0,94	66,36	0,72	69,62	0,71	69,75	No	Dropout	30	RMSProp
12,10	24,90	12,03	25,35	12,09	25,00	Si	Dropout	30	SGD

Tabla 5. Experimento 2: CNN desde cero con Aumento de datos

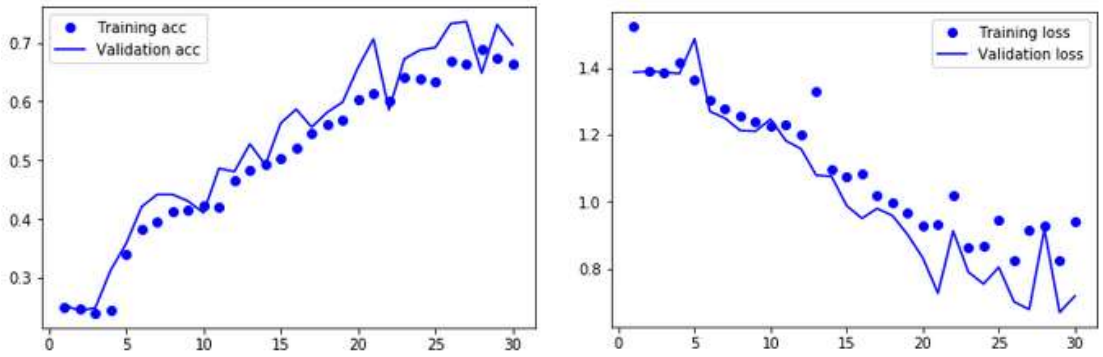


Figura 20. Experimento 2**.** Gráfica Exactitud y pérdida. Con optimizador "RMSProp"

Podemos ver dos efectos con el aumento de datos: una mejora de la exactitud en la clasificación, de 63,50% a **69,75%** con aumento de datos, y se evita en gran medida el sobreajuste, ya que se puede ver que la línea de puntos de entrenamiento se acerca al camino seguido por los de validación. Por otro lado, seguimos evidenciando que el optimizador SGD no es el adecuado.

6. EXTRACCIÓN DE CARACTERÍSTICAS CON MODELOS PREENTRENADOS

Una técnica más avanzada que la anterior, es el uso de modelos de CNN preentrenados en lugar de entrenar una CNN desde cero. Una CNN preentrenada, es un modelo de red neuronal convolucional que fue entrenada con un conjunto de datos muy amplio, en el que las jerarquías de patrones aprendidos generalizan bien para otros conjunto de imágenes con pocos datos *[5]*. Las redes preentrenadas que se utilizan en estos experimentos, están preentrenadas con el conjunto de imágenes "**ImageNet**", formadas principalmente por clases de imágenes de animales y objetos cotidianos, además contiene muchas subclases diferentes de las imágenes, por ejemplo, diferentes clases de perros, dentro de la clase "perro".

Los modelos utilizados, están disponibles gracias a sus desarrolladores de forma gratuita para su uso, y desde las librerías **Keras** para Python pueden importarse fácilmente *[14]*. La técnica se basa en que las CNN, están formadas por los bloques convolucionales que transforman los tensores de entrada hasta la capa de salida que incluye un clasificador. El clasificador siempre es específico al conjunto de datos que queremos clasificar, y la parte convolucional, a su entrada es más generalista y a medida que se acerca al clasificador es más específica al conjunto de datos que se entrena. Por lo tanto, interesa utilizar la parte convolucional de un modelo preentrenado (conjunto de pesos), añadir un clasificador propio, y entrenar el conjunto con nuestros datos (***Figura 21***). Para obtener el conjunto de pesos del modelo preentrenado, se debe aplicar un algoritmo **de extracción de características**.

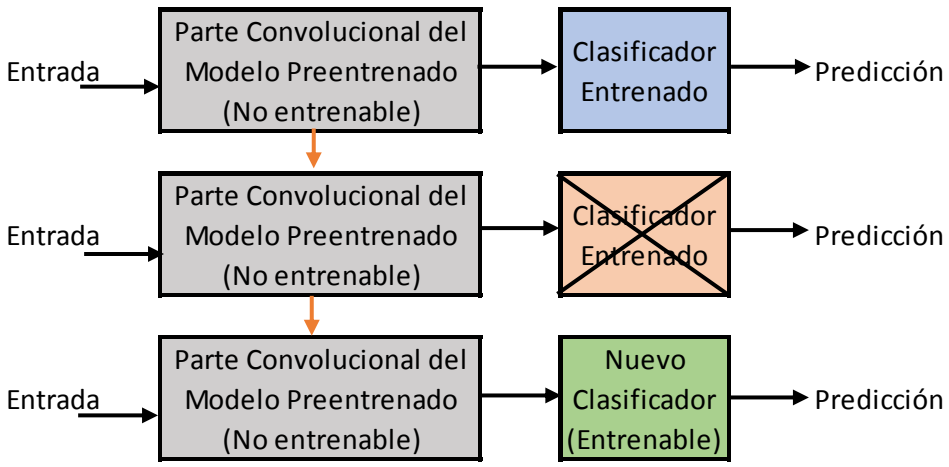


Figura 21. Entrenamiento de modelo preentrenado con nuevo clasificador

En la ***tabla 6*** se muestra el tamaño de entrada estándar de las imágenes que se han utilizado para cada modelo según *[13]*.

Entrada	Modelo Preentrenado	Estructura
224 x 224	VGG16	*[19]*
224 x 224	VGG19	*[20]*
224 x 224	ResNet50	*[20]*
299 x 299	Inception V3	*Figura 25 [16]*
299 x 299	Xception	*[24]*
299 x 299	Inception ResNet V1	*[21]*
299 x 299	Inception ResNet V2	*[22]*
299 x 299	Inception V4	*[23]*

Tabla 6. Tamaño de imágenes de entrada estándar para los modelos y estructura

Los algoritmos creados son adaptaciones de (*Cap. 5.3 [5]*) y *[14]*, lo pasos a aplicar son los siguientes:

1. Carga del conjunto de datos, haciendo referencia a los directorios: train, validation y test
2. Cargar sólo la parte convolucional del modelo preentrenado con el conjunto de pesos de ImageNet
3. Preprocesado de imágenes de entrada y generación aleatoria de imágenes
4. Extracción de características del modelo preentrenado con nuestro conjunto de datos
5. Crear un clasificador y unirlo al final de la parte convolucional del modelo preentrenado
6. Compilar el modelo completo: modelo preentrenado con el nuevo clasificador
7. Entrenar el modelo con los datos de entrenamiento (train) y validarlo con los datos de validación (validation)
8. Mostrar gráficas de efectividad (*Accuracy*) y pérdida (*Loss*)
9. Evaluar el modelo con los datos de test

6.1. Experimento 3. Extracción de características rápido

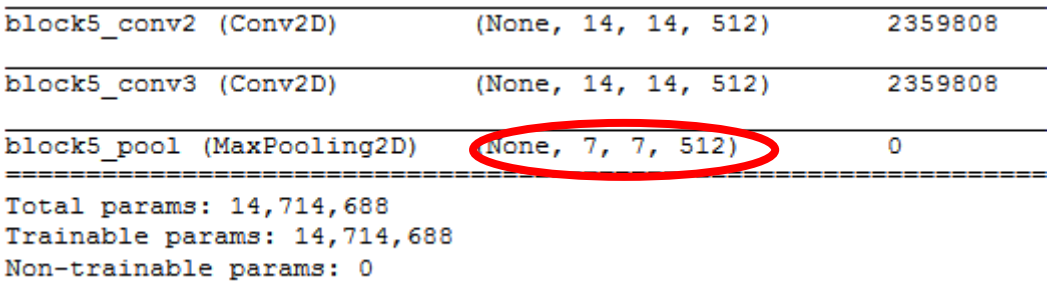
En este experimento se realizan las primeras pruebas de extracción de características en modelos preentrenados sin guardar en disco los pesos y obtener así, una visión general de qué modelos "aprenden" con resultados aceptables.

A continuación se muestra el código (para cargar el modelo ***VGG16*** sin el clasificador, es decir, con sólo los bloques convolucionales (*include_top* = False) para el conjunto de datos **ImageNet** (*weights* = 'imagenet'). En *input_shape*, se define el tamaño de entrada estándar del modelo 229 x 229 x 3.

```
from keras.applications import VGG16
conv_base = VGG16(weights = 'imagenet',
                  include_top = False,
                  input_shape = (width_image, height_image, channels_image))
```

En el resto de modelos se realiza el mismo procedimiento, pero especificando en *input_shape* el tamaño de entrada concreto para el modelo utilizado, y haciendo referencia al modelo a cargar.

Utilizando el comando *model.summary(),* mostramos el modelo creado y podemos ver, el tamaño del tensor de salida de la CNN. En *VGG16* la salida corresponde a un tensor de (7, 7, 512). Por lo tanto, cuando enlacemos la parte convolucional del modelo al clasificador, deberemos indicarle al clasificador, que su entrada sea del tamaño de la salida del modelo.



Pero antes, se deben extraer las características del modelo. El siguiente código para el modelo *VGG16* extrae las características del modelo. Se debe realizar el mismo procedimiento para el resto de modelos.

```
#Extracción de características del modelo Preentrenado y nuestro dataset
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical

#Tamaño de salida de la última capa convolucional del modelo preentrenado
out_x = 7
out_y = 7
conv_len = 512
datagen = ImageDataGenerator(rescale = 1./255)

def extract_features(directory, sample_count):
    features = np.zeros(shape=(sample_count, out_x, out_x, conv_len))
    labels   =   to_categorical(np.zeros(shape=(sample_count)),nb_clases)   #INDICAR
NUMERO DE CLASES
    generator = datagen.flow_from_directory(
        directory,
        target_size = (height_image, width_image),
        batch_size = batch_size,
        #classes = 4,
        class_mode = 'categorical')
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(inputs_batch) ##Asociar al modelo
```

```
        features[i * batch_size : (i + 1) * batch_size] = features_batch
        labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels

#Train: 1000 muestras x clase, con 4 clases, 1000 x 4 = 4000
train_features, train_labels = extract_features(train_dir, nb_train)
#validation 300 muestras x clase con 4 clases, 300 x 4 = 1200
validation_features,    validation_labels    =    extract_features(validation_dir,
nb_validation)
#test 300 muestras x clase con 4 clases, 300 x 4 = 1200
test_features, test_labels = extract_features(test_dir, nb_test)
```

Una vez cargada la parte convolucional del modelo preentrenado con sus pesos y extraídas sus características, se debe añadir el clasificador sin entrenar.

```
#Definimos un clasificador densamente conectado capacitado con los datos y etiquetas
obtenidas antes

#Aplicamos capa de abandono Dropout y función de activación "sigmoid" al final
from keras import layers

top_model = models.Sequential()
top_model.add(layers.Dense(256, activation = 'relu', input_dim = out_x * out_y *
conv_len))
top_model.add(layers.Dropout(0.5))
top_model.add(layers.Dense(nb_clases, activation = 'sigmoid')) #INDICAR Nro CLASES
```

La variable *Input_shape* especifica el tamaño de entrada del clasificador, que como se comentó anteriormente corresponde a (7, 7, 512). En la capa *Dense*, el valor 256, corresponde al número de neuronas completamente conectadas, que en el caso de los modelos *VGG16* y *VGG19* son de 256.

A partir de aquí, se puede proceder a compilar el modelo y entrenarlo como se hizo en el experimento anterior. Se utiliza el optimizador *SMRProp*, y el *SGD* en los modelos *VGG16* y *VGG19*, pero al obtener deficientes resultados, se deja de aplicar *SGD* en el resto de modelos.

```
#Para la compilación usaremos el optimizador RMSprop ya que termina
#la red con una sola unidad sigmoidea

top_model.compile(loss = 'categorical_crossentropy',
              optimizer = optimizers.RMSprop(lr=2e-5),
              metrics = ['acc'])


history = top_model.fit(
    train_features,
    train_labels,
    batch_size = batch_size,
    epochs = epochs,
    verbose = 1,
    validation_data = (validation_features, validation_labels))
```

Con algoritmo rápido de extracción de características obtenemos:

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	Acc	Loss	Acc	Loss	Acc	Sobreajuste	Regulariz	Epoch	Optimizer
Modelo VGG16									
0,39	87,45	0,56	78,92	0,68	71,50	a 15 epochs	Dropout	30	RMSProp
0,25	93,03	0,50	81,67	0,56	78,33	a 15 epochs	Dropout	50	RMSProp
0,12	97,85	0,50	81,25	0,56	79,42	a 15 epochs	Dropout	80	RMSProp
0,07	98,73	0,54	81,00	0,68	74,83	a 15 epochs	Dropout	100	RMSProp
1,28	36,40	1,20	41,25	1,18	43,67	Si	Dropout	80	SGD
Modelo VGG19									
0,44	84,68	0,56	79,33	0,67	71,83	a 15 epochs	Dropout	30	RMSProp
0,29	91,65	0,51	80,92	0,65	73,75	a 15 epochs	Dropout	50	RMSProp
0,15	96,70	0,53	79,67	0,59	77,92	a 15 epochs	Dropout	80	RMSProp
0,11	98,32	0,50	80,58	0,58	77,83	a 15 epochs	Dropout	100	RMSProp
1,39	25,37	1,39	25,00	1,39	25,00	Si	Dropout	80	SGD
Modelo ResNet50									
1,38	27,70	1,38	48,08	1,38	38,50	variable	Dropout	80	RMSProp
Modelo Inception ResnNetV1									
1,39	24,72	1,39	25,00	-	25,00	a 10 epochs	Dropout	80	RMSProp
Modelo Inception ResnNetV2									
1,39	24,72	1,39	25,00	-	25,00	a 10 epochs	Dropout	80	RMSProp
Modelo Inception V3									
0,01	99,72	0,89	80,33	1,06	74,00	a 10 epochs	Dropout	80	RMSProp
Modelo Xception									
0,02	99,52	0,64	82,75	0,79	79,75	a 10 epochs	Dropout	80	RMSProp
Modelo Inception V4									
1,34	36,95	1,31	40,00	1,28	39,43	a 10 epochs	Dropout	80	RMSProp

Tabla 7. Experimento 3: Extracción de características rápida

Observando los resultados anteriores, aplicando algoritmos rápidos, se aprecia que los modelos *InceptionResNetV1*, *InceptionResNetV2* y *InceptionV4* no dan buenos resultados después de 80 épocas de entrenamiento, las CNN de los modelos no obtienen un rendimiento superior a un 25% en las *ResNet*, cuando según *[15]*, se debería obtener un rendimiento superior a *VGG16*, *VGG19* y a *InceptionV3*. Con el modelo *ResNet50* ocurre lo mismo, obtenemos deficientes resultados, ya que debería ser algo superior a los modelos *VGG16* y *VGG19*. En los modelos *VGG16* y *VGG19* obtenemos a 80 épocas una exactitud de predicción de clases de 79,42% y 77,92% respectivamente, siendo estos valores algo superiores a los obtenidos en *[15].* Inception V3 no obtiene mejores resultados que los anteriores pese a que debería ser superior, pero el modelo obtiene resultados aceptables a 80 épocas.

Por último, el algoritmo que mejor clasifica las células en este experimento rápido, es el modelo *Xception*, con una exactitud del 79,75%. Por lo tanto, se aprecia que a 80 épocas es cuando

obtenemos mejores resultados sin sobreentrenar los modelos, el optimizador *RMSProp* funciona correctamente en estos algoritmos, pero se aprecia sobreajuste aproximadamente a las 10 épocas, esto es debido a que no se está utilizando ninguna capa de regularización *Dropout*, ni aumento de datos. También se puede decir, que los modelos *VGG16*, *VGG19*, *InceptionV3* y *Xception* se les ha podido entrenar y que "aprenden".

6.2. Experimento 4. Extracción de características

En el experimento 3 se utilizaron algoritmos rápidos para cada modelo en los que no se guardaban en disco las configuraciones de pesos, en este experimento, se modifican los algoritmos para guardar estos pesos en disco y que puedan ser reutilizados.

Después de observar los deficientes resultados en los modelos tipo *Inception*, y sabiendo que éstos deberían ofrecer mejores resultados, se investigaron las causas, y se observó que éstos modelos funcionan mejor con un clasificador no secuencial. Por lo tanto, en los modelos tipo *Inception*, se realizan pruebas con los dos clasificadores. La salida del modelo *InceptionV3* es:

```
activation_94 (Activation)       (None, 8, 8, 192)     0

mixed10 (Concatenate)            (None, 8, 8, 2048)    0


==================================================================
Total params: 21,802,784
```

Y la del modelo Xception es:

```
block14_sepconv2 (SeparableConv (None, 10, 10, 2048) 3159552

block14_sepconv2_bn (BatchNorma (None, 10, 10, 2048) 8192

block14_sepconv2_act (Activatio (None, 10, 10, 2048) 0
==================================================================
Total params: 20,861,480
```

Por lo tanto, el clasificador deberá tener como entrada un tensor de esos tamaños.

El clasificador no secuencial está formado por: una capa **GlobalMaxPooling2D** que aplica agrupamiento máximo a las ramificaciones de la CNN. En el modelo InceptionV3 el clasificador tiene como entrada, el especificado en la salida del modelo (8, 8, 2048).

```
def add_new_last_layer(base_model, nb_classes):
    FC_SIZE =1001
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(FC_SIZE, activation='relu')(x) #new FC layer, random init
```

```
if class_mode == "binary":
    predictions = Dense(1, activation='sigmoid')(x) #new sigmoid layer
else:
    predictions = Dense(nb_classes, activation='softmax')(x) #new softmax layer

model = Model(inputs=base_model.input, outputs=predictions)
return model
```

Después de cargar el modelo a estudiar, extraer las características, y enlazar la parte convolucional del modelo con el clasificador, se debe compilar el modelo y entrenarlo.

Resultados obtenidos en el **Experimento 4:**

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	Acc	Loss	Acc	Loss	Acc	Sobreajuste	Regulariz	Epoch	Optimizer
Feature Extraction Disc: Model VGG16									
0,10	98,30	0,50	81,00	0,57	79,17	a 10 epochs	Dropout	80	RMSProp
0,02	99,88	0,60	81,17	0,78	75,00	a 10 epochs	Dropout	80	RMSProp
0,00	100,00	0,97	80,17	1,09	78,92	a 10 epochs	Dropout	300	RMSProp
0,09	98,38	0,50	81,25	0,57	78,92	a 10 epochs	Dropout	80	RMSProp
Feature Extraction Disc: Model VGG19									
0,13	97,72	0,54	79,00	0,60	76,67	a 10 epochs	Dropout	80	RMSProp
0,04	99,65	0,65	78,42	0,75	75,00	a 10 epochs	Dropout	80	RMSProp
0,14	96,88	0,49	0,81	0,60	76,25	a 10 epochs	Dropout	80	RMSProp
Feature Extraction Disc: Model ResNet50									
1,34	39,82	1,34	47,58	1,35	42,33	Si	Dropout	80	RMSProp
1,17	50,60	1,15	56,00	1,20	45,83	variable	Dropout	500	RMSProp
Feature Extraction Disc: Model InceptionV3									
0,03	99,20	0,73	81,17	0,93	77,50	Si	Dropout	30	RMSProp
0,01	99,72	1,04	80,50	1,20	78,50	Si	Dropout	60	RMSProp
0,24	93,77	0,43	83,83	0,53	79,42	a 15 epochs	No Drop	90	RMSProp
Feature Extraction Disc: Model Xception									
0,00	99,95	0,80	85,67	1,09	81,25	Si	Dropout	80	RMSProp
0,04	98,88	0,55	84,75	0,83	79,25	Si	Dropout	30	RMSProp
0,01	99,82	0,79	83,75	1,38	74,50	Si	Dropout	60	RMSProp
0,01	99,80	0,82	85,42	1,06	82,00	Si	Dropout	90	RMSProp
0,27	91,17	0,41	84,33	0,46	82,50	a 20 epochs	No Drop	90	RMSProp
Feature Extraction Disc: Model InceptionV4									
0,73	71,47	0,85	67,83	1,13	67,53	Si	Dropout	5000	RMSProp
0,57	78,12	0,82	71,75	0,86	64,25	Si	Dropout	5000	RMSProp
0,72	71,50	0,85	67,58	0,92	60,33	No	Dropout	80	RMSProp
0,69	72,92	0,82	69,25	0,91	61,42	poco	Dropout	5000	RMSProp

Tabla 8. Experimento 4: Extracción de características en disco

En este experimento, se han entrenado los modelos a 80 épocas que parecía dar buenos resultados en los anteriores experimentos, y además, se ha aumentado el número de épocas en algunas pruebas para comprobar si existe mejora en los resultados.

Todos los modelos han obtenido mejoras respecto a los anteriores experimentos, excepto en los modelos *VGG16* y *VGG19*. El modelo *ResNet50* ha mejorado, pero no con rendimiento satisfactorio, por lo que no se utilizará en el siguiente experimento.

Los modelos *InceptionV3* y *V4* obtienen rendimientos mejorados, y por lo general, aumenta el rendimiento al aumentar el número de épocas de entrenamiento teniendo en cuenta la aleatoriedad a la hora de generar datos, y que en cada ejecución de algoritmos se obtienen resultados no iguales. Tampoco se han observado diferencias significativas al utilizar un clasificador secuencial o no en los modelos tipo *Inception*.

El modelo *InceptionV4* ha mejorado respecto a los anteriores experimentos, pero con rendimiento bajo. Este modelo no se incluye por defecto en las librerías keras, por lo que, se ha implementado la CNN a través de código y no importando el modelo. El bajo rendimiento, se intuye, a que el conjunto de pesos cargado no tenga el número de entrenamientos que los modelos incluidos por defecto en keras, ya que para obtener un 67,53% de eficacia, se ha entrenado el modelo a 5000 épocas.

El modelo en este experimento que presenta mejores resultados es Xception, con un 82,50% de exactitud en las predicciones de los datos de test.

7. FINE TUNING EN MODELOS PREENTRENADOS

El **Fine Tuning** o Ajuste Fino, es la técnica más sofisticada utilizada en este trabajo, ya que utilizando modelos preentrenados complementa a la extracción de características vista anteriormente. El Fine Tuning consiste en descongelar algunas capas cercanas a la salida de la parte convolucional del modelo preentrenado utilizado para la extracción de características y en el entrenamiento del clasificador nuevo, y después entrenar todo el conjunto con nuestros datos. (*Cap. 5.3.2 [5]*).

En los experimentos 3 y 4 en el que se utilizan modelos preentrenados, la parte convolucional de los modelos preentrenados estaban "congelados", es decir, no aprendían, o lo que es lo mismo, no ajustaban sus pesos sólo el clasificador los ajustaba. En la parte convolucional se cargaban los pesos del conjunto de datos ImageNet.

Con el Fine Tuning se descongelan algunas capas cercanas al clasificador para que se ajusten los pesos al conjunto de datos de las células.

Los pasos a seguir en los algoritmos, aprovechando la extracción de características del experimento 4 para cada modelo son:

1. Unir el clasificador nuevo a la parte convolucional del modelo preentrenado.
2. Congelar la parte convolucional del modelo preentrenado
3. Entrenar el clasificador
4. Descongelar algunas capas del modelo preentrenado.
5. Entrenar la parte convolucional del modelo preentrenado con las capas cercanas al clasificador descongeladas unidas al clasificador.

Los pasos del 1 al 3, se realizaron en el experimento 4 y se guardaron en disco. En este experimento, se cargan los datos guardados (**transferencia de aprendizaje**) y se prosigue desde el paso 4. Los algoritmos están basados en *[14]*.

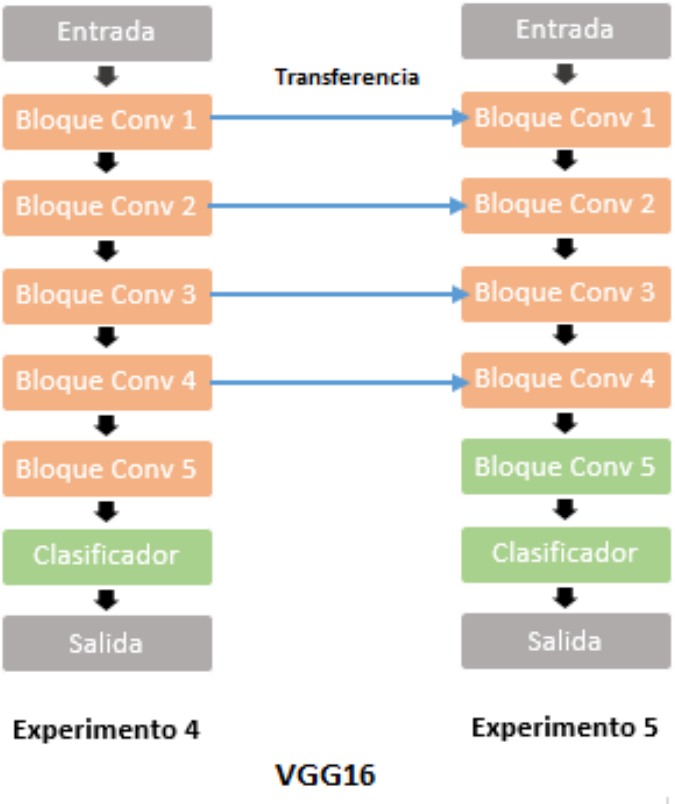
Los modelos que se utilizan para Fine Tuning son los que obtuvieron mejores resultados en los experimentos anteriores: *VGG16*, *VGG19*, *InceptionV3*, *InceptionV4* y *Xception*.

Los modelos preentrenados tienen arquitecturas diferentes, por ejemplo, los modelos *VGG16* y *VGG19* **son modelos secuenciales**, es decir, tienen bloques consecutivos de capas convolucionales seguidas de la capa de agrupamiento parecido a la CNN creada desde cero de los experimentos 1 y 2. En cambio los modelos *InceptionV3*, *V4* y *Xception*, están formados por módulos ***Inception***, que **no**

son secuenciales en sí mismos *[16]*. Por lo tanto, el clasificador de los modelos secuenciales y el de los modelos *Inception* pueden ser diferentes, dependiendo de dónde se comiencen a descongelar las capas. Es por ello, que en los siguientes experimentos se tratan de forma separada.

7.1. Experimento 5. Fine Tuning modelos VGG

En la zona izquierda de la imagen (el experimento 4), los bloques convolucionales del modelo *VGG* se enlazaron a un clasificador completamente conectado y se guardó la configuración de los pesos. En este experimento (derecha de la imagen), se **transfieren** los pesos guardados anteriormente al modelo preentrenado con el clasificador, y se **descongela** el último bloque convolucional para entrenarlo conjuntamente con el conjunto de datos de las células.



El modelo *VGG16* consta de cinco bloques convolucionales (**Bloque_Conv**) unidos de forma secuencial. Cada uno de ellos está compuesto por dos capas convolucionales *Conv2D*, seguida de una capa *MaxPooling2D* al que se le añade un clasificador secuencial como se vio en el experimento 3 y 4. (*Apartado* 6.1 y 6.2).

El modelo *VGG19* básicamente es igual a *VGG16*, pero los bloques convolucionales 3, 4 y 5, en lugar de tener dos capas *Conv2D* tienen cuatro, seguidas de la capa *MaxPooling2D* y el mismo clasificador secuencial a la salida que VGG16.

Una vez se ha cargado el modelo preentrenado (*VGG16* o *VGG19*) con los pesos de *ImageNet*, enlazamos con el clasificador secuencial, teniendo en cuenta que el tensor de entrada del clasificador, deber ser del tamaño del tensor de salida del modelo preentrenado. El tamaño del tensor de salida en la parte convolucional de ambos modelos es de (7, 7, 512). Después se procede a cargar los pesos guardados en la extracción de características.

```
# top model
top_model = Sequential()
top_model.add(Flatten(input_shape=base_model.output_shape[1:]))
top_model.add(Dense(256,  activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(nb_classes,  activation='softmax'))

# base model has its weights, now we load the weights on the top layer
top_model.load_weights("VGG/VGG16_FE_model_1.h5")

# we join base and top it has to be updated to api2
model_total        =        Model(input    =    base_model.input,        output    =
top_model(base_model.output))
```

Se descongela el último bloque convolucional del modelo preentrenado:

```
# freezing layers implies they will not update their weights over the training
for  layer  in  model_total.layers[:4]: #15
    layer.trainable = False
```

En el preprocesado se utiliza el aumento de datos visto en el experimento 2 (*Apartado 5.2*) y se procede a compilar, pero esta vez con el optimizador *SGD* y una tasa de aprendizaje de 0.0001 (aprendizaje lento) en lugar de 0.001. Esto es debido a que en Fine Tuning no es deseable que se actualicen los pesos a grandes saltos, sino que, los pesos se actualizan teniendo en cuenta el gradiente del paso anterior, y no de un grupo de ellos. (*Cap. 11 p. 294 [4]*)

```
#Compile model
model_total.compile(optimizer=SGD(lr=1e-4,  momentum=0.9),
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])
```

Después de 60 épocas de entrenamiento y descongelando 4 capas para cada modelo *VGG16* y *VGG19*, se obtiene:

TRAIN		VALIDATION		TEST		balanced_dataset				
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer	
Fine Tuning: Model VGG16							**NoteBook: E_Fine_Tuning_VGG16**			
0,35	86,77	0,32	88,58	0,34	87,58	No	Dropout	60	SGD	4
Fine Tuning: Model VGG19							**NoteBook: E_Fine_Tuning_VGG19**			
0,29	89,14	0,26	88,83	0,27	89,92	No	Dropout	60	SGD	4

Tabla 9. Experimento 5: Fine Tuning modelo VGG16 y VGG19

En los dos casos se han obtenido mejoras respecto a los experimentos anteriores. En el modelo *VGG19* se ha mantenido congelada una capa convolucional del último bloque y en *VGG16* se ha descongelado completamente él último bloque convolucional.

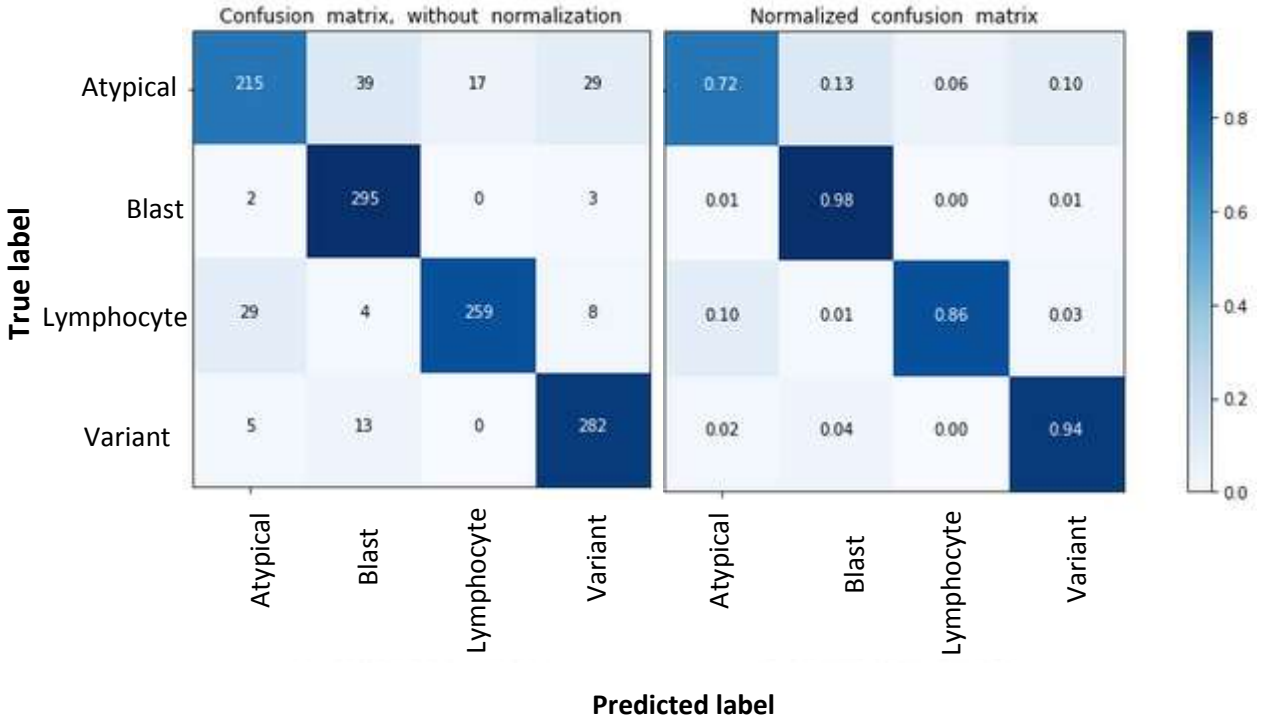


Figura 22. Experimento 5. Matriz de confusión para Fine Tuning modelo VGG16

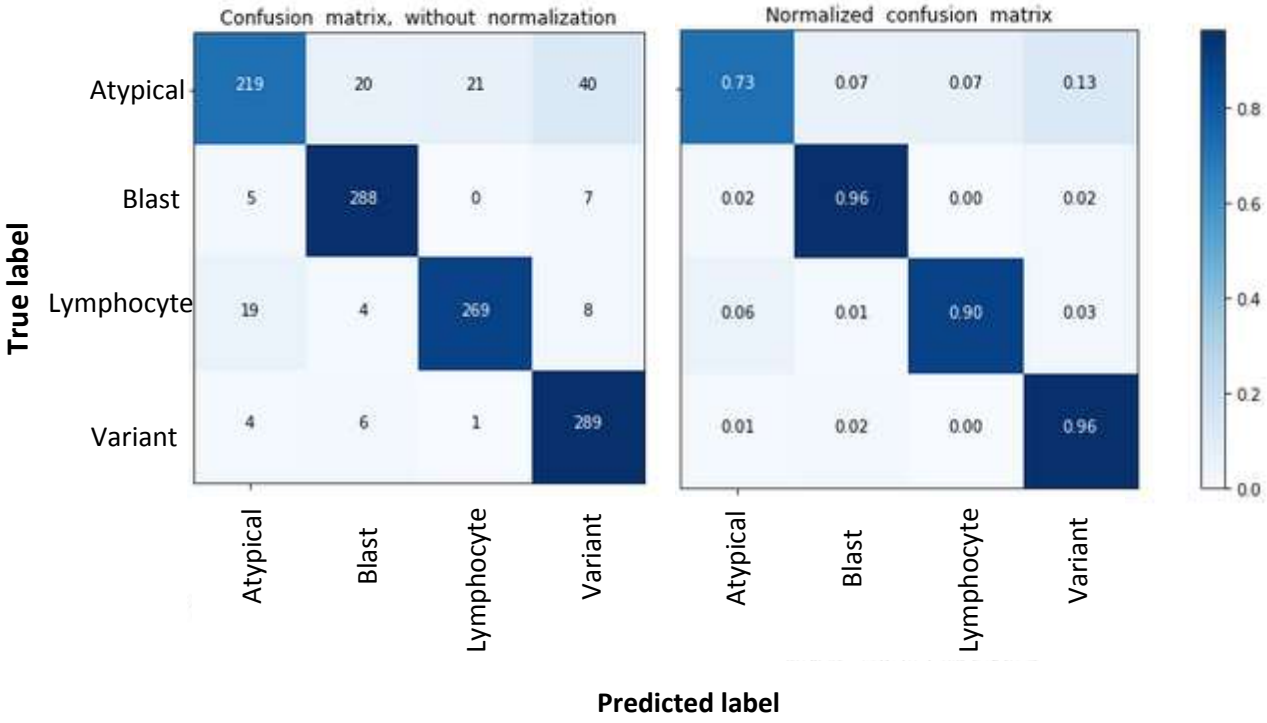


Figura 23. Matriz de confusión para Fine Tuning modelo VGG19

Para interpretar correctamente las matrices de confusión, se debe observar la diagonal que indica el número de imágenes acertadas correctamente de cada clase. El resto son los errores de clasificación. Cada columna y fila corresponde a cada una de las clases, es decir, ATYPICAL_LYMPHOCYTE, BLAST, LYMPHOCYTE y VARIANT_LYMPHOCYTE. Por ejemplo, en la primera fila de la imagen de la izquierda se tienen 215 imágenes clasificadas correctamente como ATYPICAL_LYMPHOCYTE, y erróneamente 39 imágenes como BLAST, 17 imágenes como LYMPHOCYTE y 29 imágenes como VARIANT_LYMPHOCYTE, así para cada fila. En la imagen de la derecha, se normalizan los mismos resultados en porcentaje.

7.2. Experimento 6. Fine Tuning modelos Inception

En los modelos Inception se procede de forma semejante a los modelos *VGG*, pero teniendo en cuenta que la estructura de los bloques convolucionales es diferente. Para comprender la estructura de estas CNN, a continuación se muestra un módulo o bloque *Inception*.

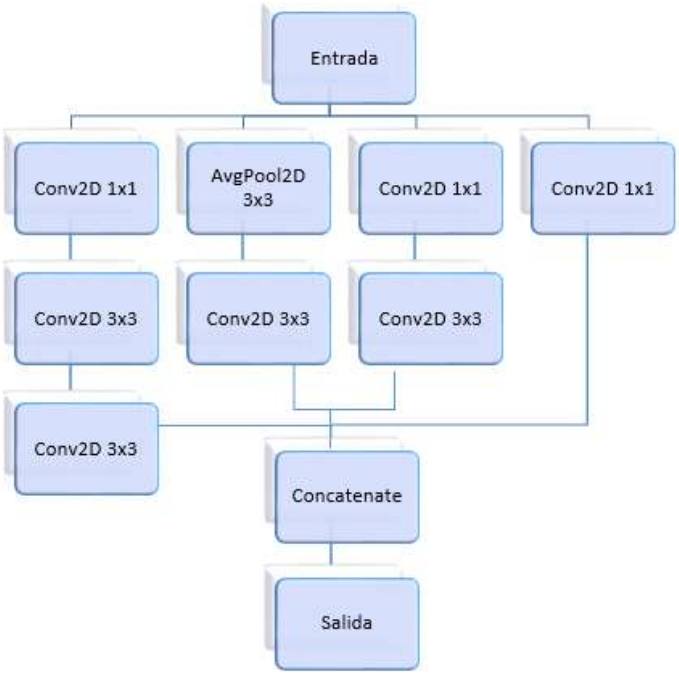


Figura 24. Esquema módulo Inception.

Si descongelamos uno de los módulos *Inception* (***Figura 24***) en alguna capa entre su entrada de módulo o salida, se debe concatenar la salida de esa rama con las otras tres. Ya que a partir de la entrada se ramifica en cuatro. A continuación, se muestra la complejidad de la CNN *InceptionV3*.

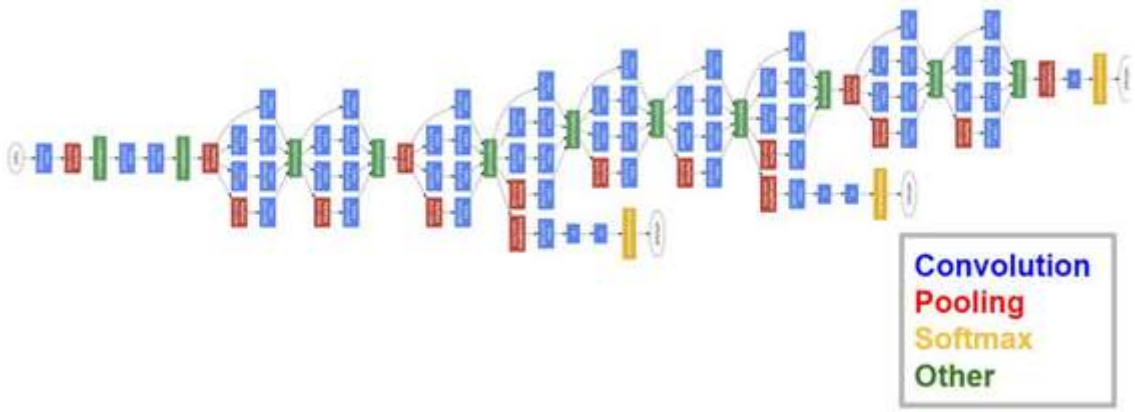


Figura 25. Esquema del modelo Inception V3. *[16]*

Como se puede ver, en estos modelos no es tan evidente en qué parte de la CNN congelar las capas.

En unas primeras pruebas, se reutilizó el código para Fine Tuning de los modelos *VGG*, descongelando las cuatro últimas capas de la parte convolucional del modelo Inception V3, y utilizando un clasificador secuencial. Los resultados no fueron satisfactorios, ya que la CNN no aprendía, obteniendo unos resultados de exactitud en test del 25%. Teniendo cuatro clases de células, esto equivaldría a lanzar una moneda para realizar las predicciones (25% X 4=100%), lo que es totalmente inaceptable. Para solucionar este problema se probó cambiando los tipos de optimizadores: *RMSProp*, *SGD*, *Adam*, *Adagrad* y *Nadam* (***Tabla 10***).

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	**Acc**	**Loss**	**Acc**	**Loss**	**Acc**	**Sobrea**	**Regulariz**	**Epoch**	**Optimizer**
Fine Tunning: Model Inception V3									
12,04	25,30	12,05	25,26	12,09	25,00	Si	Dropout	1/5	RMSProp x 2
12,10	24,95	12,05	25,26	12,09	25,00	Si	Dropout	1/5	RMSP + SGD
12,10	24,90	12,05	25,26	12,09	25,00	Si	Dropout	1/5	RMSP + Adam
12,09	24,98	12,05	25,26	12,09	25,00	Si	Dropout	1/5	RMSP + Adagrad
12,06	25,18	12,05	25,26	12,09	25,00	Si	Dropout	1/5	RMSP + Nadam

Tabla 10. **Experimento 6:** Fine Tuning con Inception V3 clasificador secuencial y diferentes optimizadores

A partir de estos resultados, se pudo comprobar que descongelar cuatro capas de un modelo *Inception* no es adecuado, ya que, como se vio en el módulo *Inception* la red está ramificada.

Se decide probar otra técnica en el modelo *InceptionV3*, congelando 172 capas según *[17]* y con el clasificador secuencial. Pero no se obtienen buenos resultados (***Tabla 11***), por lo que se prueba congelando 249 capas según *[18]*, y se obtienen ligeramente mejores resultados, pero inaceptables.

TRAIN		VALIDATION		TEST		balanced_dataset			
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer
Fine Tuning: Model Inception V3									
0,38	86,31	8,77	38,63	8,80	38,67	Si	Dropout	30	RMSProp
0,61	77,40	6,28	39,32	6,28	39,33	Si	Dropout	30	RMSProp

Tabla 11. Experimento 6: Fine Tuning modelo InceptionV3 con clasificador secuencial

Después de realizar pruebas con los optimizadores, y modificando el número de capas congeladas y no obteniendo buenos resultados, se modifica el clasificador secuencial por **el no secuencial** que es el que le corresponde a las CNN basadas en módulos *Inception*.

En estas pruebas con Inception V3, se realiza una prueba congelando 172 capas y dos más congelando 249 capas, con clasificador no secuencial. En estas pruebas podemos ver mejores resultados, pero siguen sin mejorar (***Tabla 12***).

TRAIN		VALIDATION		TEST		balanced_dataset				
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer	
Fine Tuning: Model InceptionV3										
0,19	92,97	5,42	31,16	5,42	31,42	Si	Drop.	30	RMSP	249
0,63	76,69	2,78	37,85	2,80	37,33	Si	Drop	3	RMSP + SDG	249
0,55	79,08	2,27	31,42	2,26	31,50	Si	Drop.	30	RMSP + SDG	172

Tabla 12. Experimento 6: Fine Tuning en Modelo Inception V3 clasificador no secuencial

Con estos resultados inferiores a la extracción de características, es evidente que existe algún error en los algoritmos. Después de consultar las funciones de pérdida y las funciones de activación (*p. 114 [5]*), se comprueba que se está utilizando una función de activación incorrecta. Por lo que se decide cambiar la función de activación *Sigmoid* por la función de activación *Softmax* para clasificadores multiclase. Por lo tanto, para los tres modelos: Inception V3, Inception V4 y Xception se congelan 249 capas para dejar el resto como entrenables, se utiliza el clasificador no secuencial SGD con tasa de aprendizaje lenta, (A*partado 7.1)* y función de activación *Softmax (Apartado 4.3.2)*.

A continuación se muestran los resultados:

TRAIN		VALIDATION		TEST		balanced_dataset				
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer	
Fine Tuning: Model InceptionV3						E_Fine_Tuning_InceptionV3				
6,01	89,50	6,06	89,2	5,97	88,75	No	No	30	SGD	172
0,08	97,40	0,13	94,59	0,16	93,75	No	No	50	SGD	249

Tabla 13. Experimento 6: Fine Tuning en Modelo Inception V3

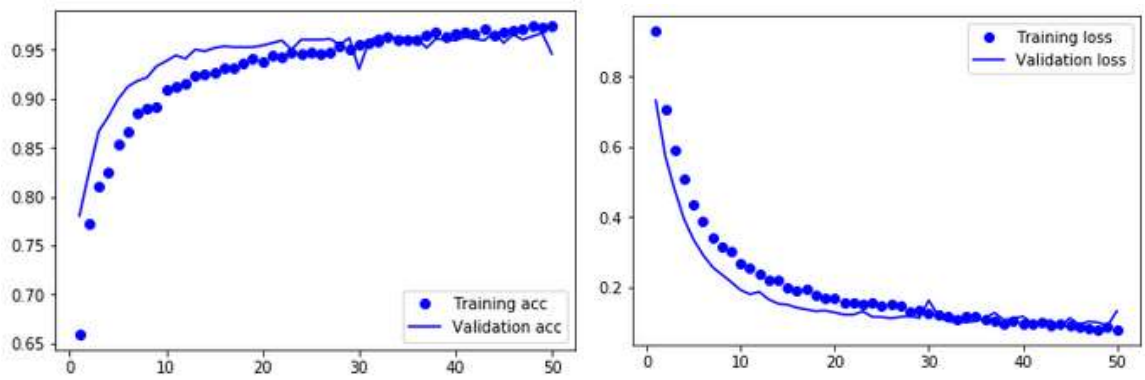


Figura 26. Exactitud y pérdida del modelo Inception V3 con 93,75% de exactitud

TRAIN		VALIDATION		TEST		balanced_dataset				
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer	
Fine Tuning: Model InceptionV4						E_Fine_Tuning_InceptionV4				
0,29	89,95	0,30	89,68	0,24	90,63	No	No	20	SGD	249
0,42	90,85	0,36	92,91	0,38	91,25	No	No	30	SGD	249
0,60	89,02	0,52	92,06	0,56	90,00	No	No	50	SGD	249

Tabla 14. Experimento 6: Fine Tuning en Modelo Inception V4

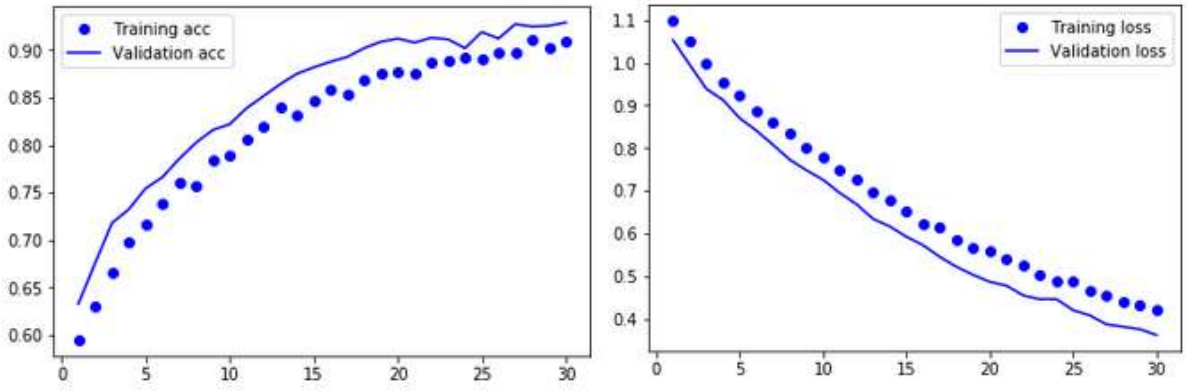


Figura 27. Exactitud y pérdida del modelo Inception V4 con 91,25% de exactitud

TRAIN		VALIDATION		TEST		balanced_dataset				
Loss	Acc	Loss	Acc	Loss	Acc	Sobreaj	Regulariz	Epoch	Optimizer	
Fine Tuning: Model Xception						NoteBook: E_Fine_Tuning_Xception				
0,19	93,05	0,19	93,24	0,12	96,25	No	No	50	SGD	249
0,13	96,00	0,14	94,51	0,14	94,50	No	No	70	SGD	249

Tabla 15. Experimento 6: Fine Tuning en Modelo Xception

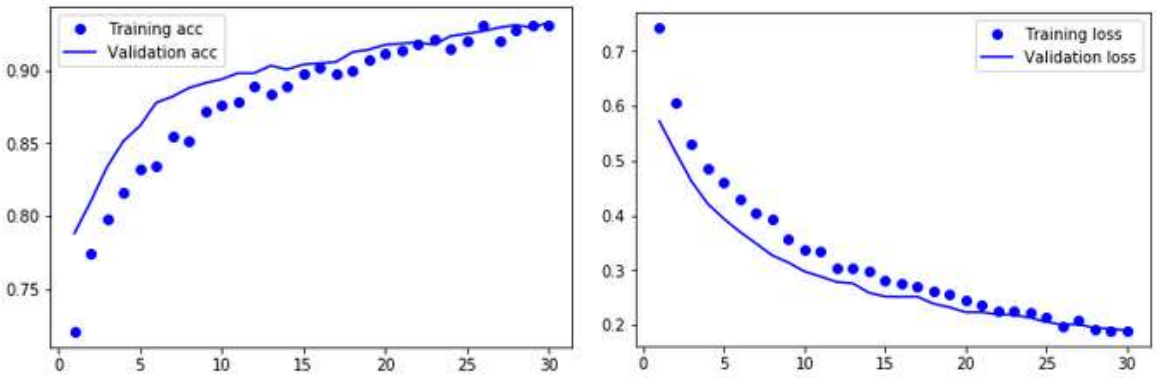


Figura 28. Exactitud y pérdida del modelo Inception V4 con 91,25% de exactitud

En los tres modelos no se observa apenas sobreajuste y los resultados son satisfactorios. Los modelos *Inception* obtienen los mejores resultados con Fine Tuning superando a los modelos secuenciales *VGG* y superando al resto de técnicas, tal y como se describe en las exactitudes obtenidas en los modelos preentrenados con el conjunto de datos *ImageNet*. *[15]*

7.2.1. Como aprende una CNN

Para comprender un poco mejor como aprende una CNN, se muestran los 16 primeros filtros aplicados a la primera capa del primer y último bloque convolucional de la CNN *InceptionV3* del experimento 6 (***Figura 29 y 30***).

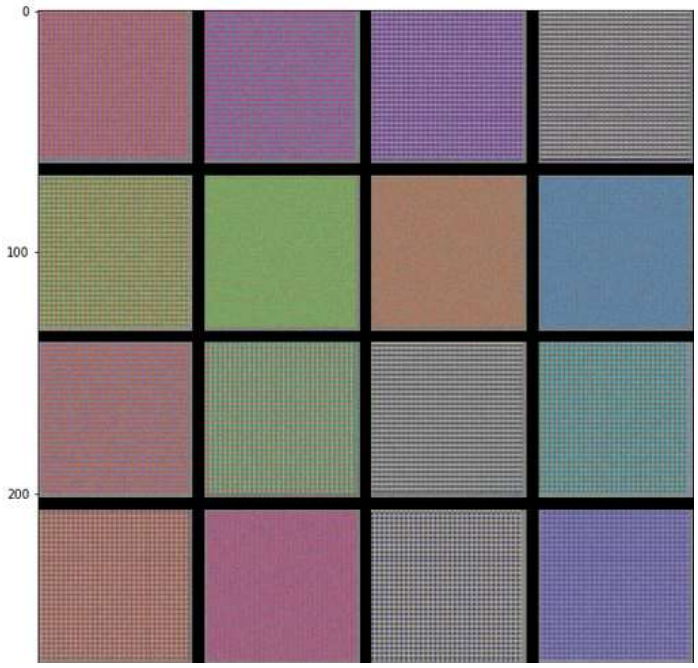


Figura 29. Los 16 primeros filtros de la 1ª. capa del 1er. bloque convolucional de Inception V3

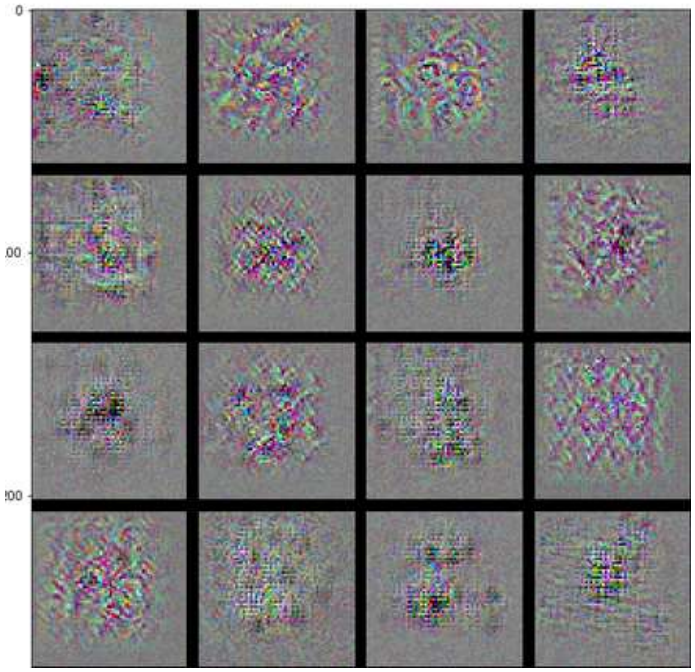


Figura 30. Los 16 primeros filtros de la 1ª. capa del último bloque convolucional de Inception V3

Estas dos visualizaciones de filtros, dicen mucho de cómo las capas convolucionales del modelo ven el "mundo". Cada capa de un bloque convolucional aprende una colección de filtros, de modo que sus entradas se pueden expresar como una combinación de filtros. Los primeros filtros del primer bloque convolucional son más generalistas y a medida que avanzan por la CNN se vuelven cada vez más complejos y refinados.

En la ***figura 29*** los filtros codifican bordes direccionales simples, bordes coloreados y colores. En cambio, en la ***figura 30*** los filtros codifican texturas y formas más complejas.

También se puede ver qué respuesta de activación devuelve cada capa cuando se la alimenta con una imagen. Por ejemplo, utilizando una imagen aleatoria del conjunto de datos (***Figura 30***), se traza una visualización completa de todas las activaciones de la red, en la que se extrae y visualiza cada canal uno al lado del otro en cada uno de los mapas de activación (***Figura 31***).

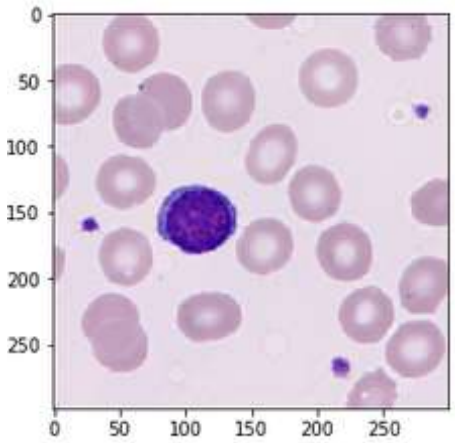


Figura 31. Imagen obtenida al azar del conjunto de datos.

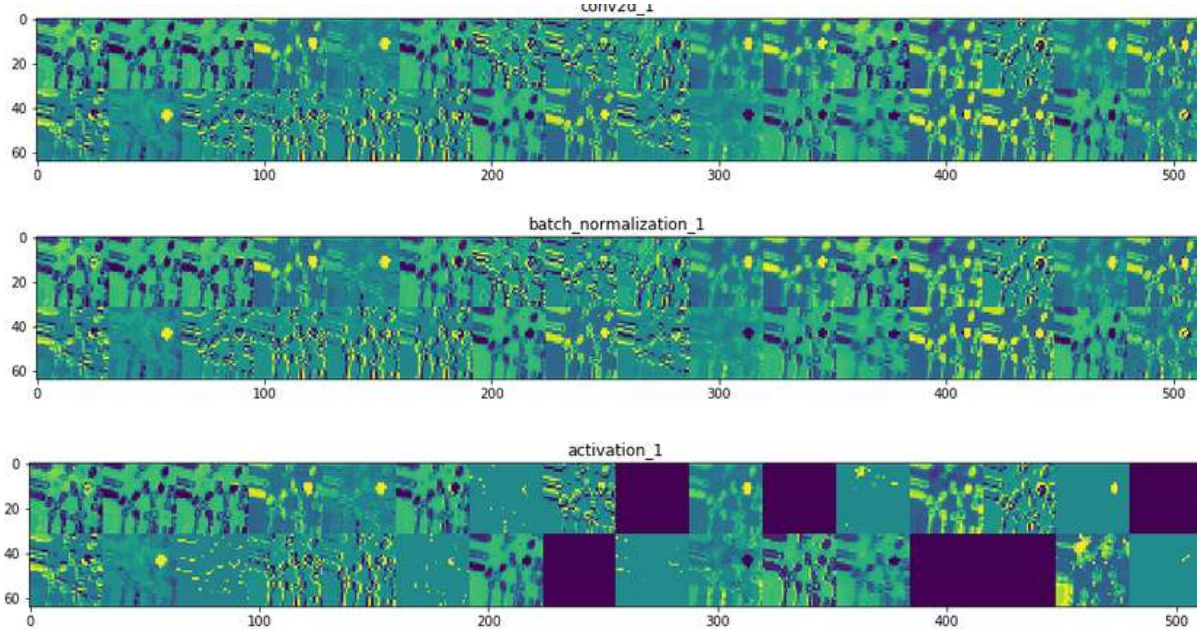


Figura 32. Activaciones en respuesta a una imagen de entrada aleatoria en modelo VGG16

En la gráfica superior de la ***figura 32***, la primera capa actúa como una colección de varios detectores de bordes. Las activaciones retienen casi toda la información de la imagen de entrada. Vemos que todos los filtros están activados. A medida que se avanza en las capas, la información se aprecia más abstracta y menos interpretable visualmente, ya que comienza a codificar conceptos de nivel superior, como puede ser la rugosidad o contorno de la célula. En la gráfica inferior, se aprecian las amplitudes de las activaciones ampliadas y no todos los filtros se activan (zonas oscuras), lo que significa que no ha encontrado el filtro en la imagen de entrada.

8. ANÁLISIS DEL IMPACTO AMBIENTAL

El algoritmo de detección y clasificación en su uso normal tiene como utilidad diferenciar patologías graves como linfomas o leucemias y derivar a la persona afectada a un centro especializado donde se le realicen las pruebas oportunas tales como, estudios genéticos y aquellos que permitan definir un diagnóstico y tratamiento temprano. Nunca debe utilizarse como sustitutivo de personal médico o para diagnóstico de enfermedades concretas, ya que no permite distinguir entre subgrupos de linfomas o leucemias.

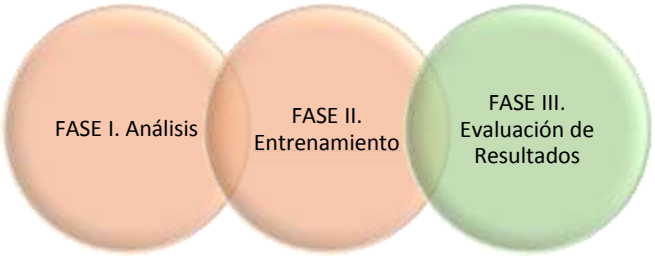
El algoritmo debería tener un seguimiento de validación, en el que se permita comprobar si las clasificaciones realizadas entre los cuatro grupos realmente fueron clasificadas correctamente, comparando el diagnóstico final de los especialistas, con el que obtuvo el algoritmo y corregir deficiencias y/o realizar mejoras. En el supuesto de que las clasificaciones del algoritmo incurran en errores significativos de clasificación, debe suspenderse su uso a la espera de revisión.

Este trabajo no genera ningún residuo o impacto contra el medio ambiente.

Diagnóstico	Económico	Tiempo
Detección temprana de patologías graves	Reducción de costes en analizadores celulares	Reducción del tiempo de dedicación
Aplicación de tratamiento médico más temprano	Reduce el número de especialistas médicos	Análisis rápido

Tabla 16. Impactos directos del proyecto

9. CONCLUSIONES



9.1. Resumen de resultados

La **Tabla 17** muestra un resumen de los mejores resultados del total de experimentos.

Técnica y modelo	TEST	
	Loss	Acc
CNN desde 0	1,22	63,50
CNN desde 0 con aumento de datos	0,71	69,75
Extracción de características VGG16	0,56	79,42
Extracción de características VGG19	0,59	77,92
Extracción de características InceptionV3	1,21	79,50
Extracción de características Xception V3	0,46	82,50
Extracción de características InceptionV4	1,13	67,53
Fine Tuning VGG16	0,34	87,58
Fine Tuning VGG19	0,31	89,92
Fine Tuning InceptionV3	0,16	93,75
Fine Tuning Xception	0,15	96,25
Fine Tuning InceptionV4	0,38	91,25

Tabla 17. Mejores resultados obtenidos evaluando el conjunto de test

Los algoritmos con los mejores resultados que se han implementados para cada experimento, se han cargado en la plataforma libre **GitHub** en un directorio personal de acceso público y al que se puede acceder a través de la dirección: https://github.com/RVillarraso/Deep_Learning

9.2. Conclusiones

En los **experimentos 1** y **2** en la que se entrena una CNN desde cero, se obtienen los resultados más bajos en clasificación de 63,50% y 69,75% respectivamente. La técnica de aumento de datos nos permite comprobar que se mejoran los resultados y se reduce parte del sobreajuste. Los algoritmos para entrenar una CNN desde cero son más rápidos para CPU y nos permiten obtener una visión general de la eficiencia del conjunto de datos con una CNN y que se debería superar con técnicas más complejas.

En el **experimento 3**, se superan los anteriores resultados, extrayendo las características de modelos de redes preentrenadas. Estos algoritmos son rápidos para CPU pero no permiten el aumento de datos, por lo que no se evita el sobreajuste. Para superar estos resultados, en el experimento 4, se extraen las características y se guardan en disco para ser reutilizadas. El modelo más eficaz es *Xception* con exactitud de 79,75% en el experimento 3 y 82,50% en el **experimento 4**.

Después de solucionar los problemas en Fine Tuning, se transfiere el aprendizaje del experimento 4 a los algoritmos del **experimento 5** (VGG) y **experimento 6** (Inception), en los que se consigue mejores resultados con los modelos *Inception*, siendo el más eficiente **Xception** con una eficacia del **96,25%** en la clasificación de las células.

Se han podido implementar técnicas que permiten que no sea relevante el tamaño de las imágenes de origen y que por lo tanto, los algoritmos no dependan del método en el que se adquirieron las imágenes.

Se consiguió balancear los datos aplicando herramientas híbridas de reducción del número de muestras excesivas por clase y aumentando el número de muestras en las clases más pequeñas.

Se eliminó el exceso de ruido (exceso de fondo) en las imágenes, recortándolas sin distorsión y centrándolas en la célula a clasificar. Además se ha conseguido mitigar el sobreajuste inicial con técnicas de regularización y aumento de datos.

Por otro lado, se han evaluado diferentes técnicas de menor a mayor complejidad con modelos de CNN preentrenados, con el fin de determinar el más eficiente para la clasificación de las células objetivo.

Mediante un modelo profundo de redes neuronales convolucionales, se ha alcanzado el objetivo de clasificación de las imágenes para cuatro tipos de imágenes de células, obtenidas con un microscopio manual, con una alta exactitud.

9.3. Perspectivas futuras

Los resultados satisfactorios en la aplicación de CNN para la clasificación de glóbulos blancos, sugiere que se amplíen los experimentos a un grupo mayor de células y patologías, pudiéndose clasificar con el mismo conjunto de datos disponible hasta 18 clases. Por otro lado, recopilar más datos, permitiría ampliar aún más los estudios, no solo a glóbulos blancos, sino a otro tipo de células como los eritrocitos o las plaquetas, que afectan a patologías en los grupos de las anemias y trombosis.

Con la potencia de las CNN no sólo se puede incluir el aporte informativo de las imágenes celulares, sino que también podrían incluirse otros tipos de datos, como los síntomas principales, número de células por tipo, datos genéticos, etc., con el fin de incorporar más información para la clasificación correcta de una gama más amplia de patologías.

En este trabajo se han utilizado redes neuronales convolucionales preentrenadas importadas con las librerías Keras, pero TensorFlow que es de más bajo nivel, permite diseñar redes neuronales convolucionales con diferentes tipos de ramificaciones, o incluso combinando diferentes modelos. Esto permitiría un mayor control en todo el proceso.

10. ANÁLISIS ECONÓMICO

Para realizar este proyecto se han necesitado pocos elementos, ya que una de sus características es que sea accesible y necesite pocos recursos. Algunas herramientas utilizadas ya se suelen disponer en entornos médicos, como puede ser una computadora personal y un servidor al que acceder a los servicios sanitarios. Por este motivo, estos dos elementos no se incluyen en el análisis económico. Partiendo que el ámbito de aplicabilidad de este proyecto es en entornos médicos y cuando se dispone a realizar una evaluación de la sangre periférica, el coste de la extracción sanguínea tampoco se incluye en este estudio.

10.1. Material

Para poder realizar este trabajo, en primer lugar se necesita un microscopio óptico para aumentar las imágenes y una cámara fotográfica digital para digitalizar las imágenes de los frotis sanguíneos. Para los experimentos del 1 al 4 con una computadora personal sería suficiente, pero para los experimentos 5 y 6, es necesario acceder a un servidor con una GPU (procesador gráfico), ya que el coste computacional es elevado e inviable para CPU. Se analizan económicamente el microscopio, la cámara y la GPU.

Concepto		Importe €
Microscopio Olympus DX43	*[31]*	6450
Objetivo OPTIKA C-P8	*[32]*	936
NVIDIA Titan Xp	*[33]*	1650
	Total (Iva Incluido)	**9036**

Tabla 18. Presupuesto del material

10.2. Software

El software utilizado (*Apartado 1.4*) es de acceso libre, por lo que no tiene ningún coste.

10.3. Ingeniería

Los costes de ingeniería se componen del tiempo empleado por un ingeniero/a biomédico junior en la duración de este proyecto que es de 24 ECTS a 25h cada ECT y el tiempo empleado por el director y codirector del proyecto aproximadamente.

Concepto	Tiempo [h]	Importe h. [€]	Subtotal [€]
Director de proyecto	24	30	720
Co-director de proyecto	24	30	720
Ingeniero junior 24 ECTSx35 h	600	10	6000
IVA 21%			1562,4
		Total (IVA Incluido)	**9002,4**

Tabla 19. Presupuesto de ingeniería

10.4. Condiciones

La realización de este proyecto tiene un coste de 9002,4 euros en conceptos de ingeniería si no fuese necesario el material de la ***tabla 18***. En caso contrario, aumentaría con la compra del material, siendo un total de 18.038,4 euros.

Esta valoración económica tiene una caducidad de 3 meses a partir de su entrega a cliente. Cualquier otro gasto no especificado en las ***tablas 18*** y ***19*** no están incluidos en el presupuesto.

Para la aceptación del presupuesto se requiere el ingreso del 40% del importe de costes de ingeniería y el 100% de costes de material, antes de tres meses.

11. BIBLIOGRAFÍA

[1] E. S. Alférez Baquero, "***Methodology for Automatic Classification of Atypical Lymphoid Cells from Peripheral Blood Cell Images***", PhD. Tesis, Universitat Politècnica de Catalunya, Barcelona, 2015.

[2] M. A. Lichtman, K. Kaushansky, T. J. Kipps, J. T. Prchal, M. M. Levi, W. J. Williams. **Williams Manual de Hematología**. México: McGrawHill Interamericana, 6ta. Edición, 2011.

[3] A. Brüel, E. I. Christensen, J. Tranum-Jensen, O. Qvortrup, F. Geneser. **Histología**. Madrid: Editorial Médica Panamericana, 4ta. Edición, 2012.

[4] Géron A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow.** USA: O'Relly Media Inc, 1a. Edición, 2017

[5] Chollet F. **Deep Learning with Python.** USA: Manning Publications Co., 1a Edición, 2018

[6] Ng A., Bensouda Mourri Y., Katanforoosh K**. Deep Learning Specialization.** Internet: https://www.coursera.org/specializations/deep-learning, [25, Oct, 2017].

[7] Masip Rodó, D., Escudero Bakx G., Benítez Iglesias R., Kanaan Izquierdo S. **Inteligencia Artificial Avanzada.** Barcelona: Editorial UOC, 2016.

[8] **CS231n: Convolutional Neural Networks for Visual Recognition.** Internet: https://cs231n.github.io/classification/. [Acceso: 20, Oct, 2017]

[9] **Unsupervised Feature Learning and Deep Learning Tutorial. Softmax Regression.** Stanford University. Internet: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/, [Acceso: 9, Ene, 2018]

[10] **Unsupervised Feature Learning and Deep Learning Tutorial. Feature Extraction using Convolution**. Stanford University. Internet: http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/, [Acceso: 9, Ene, 2018]

[11] Krizhevsky A., Sutskever I., Hinton G. E., **ImageNet Classification with Deep Convolutional Neural Networks.** Internet: www.cs.toronto.edu/~fritz/absps/imagenet.pdf [Acceso: 18, Ene, 2018]

[12] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: A Simple **Way to Prevent Neural Networks from Overfitting**. [Actualizado: 15, Jun, 2014] Internet: http://jmlr.org/papers/v15/srivastava14a.html. [Acceso: 18, Enero, 2018]

[13] Herrera Gomez A., Granados García M. **Manual de Oncología**. México: McGraw Hill Interamericana, 5ª. Edición, 2013

[14] Chollet F. **Building powerful image classification models using very little data.** [Actualizado: 5, Jun, 2016] Internet: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html. [Acceso: Feb, 2018]

[15] **Documentation for individual models**. Internet: https://keras.io/applications/ [Acceso: 17, Feb, 2018]

[16] Yu F. **A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part II).** [Actualizado: 8, Oct, 2016] Internet: https://flyyufelix.github.io/2016/10/08/fine-tuning-in-keras-part2.html. [Acceso: Marzo, 2018]

[17] **How to use keras to fine-tune inception v3 to do multi-class classification**? Internet: https://stackoverflow.com/questions/45852434/how-to-use-keras-to-fine-tune-inception-v3-to-do-multi-class-classification [Acceso: Mayo, 2018]

[18] **Fine-tune InceptionV3 on a new set of classes.** Internet: https://keras.io/applications/#fine-tune-inceptionv3-on-a-new-set-of-classes [Mayo 2018]

[19] **Estructura del modelo VGG16.** Internet: https://github.com/RVillarraso/Deep_Learning/blob/master/vgg16_modified.png

[20] **Estructura del modelo VGG19.** Internet: https://github.com/RVillarraso/Deep_Learning/blob/master/VGG19.png

[21] **Estructura del modelo Inception ResNetV1.** Internet: https://github.com/RVillarraso/Deep_Learning/blob/master/Inception%20ResNet-v1.png

[22] **Estructura del modelo Inception ResNetV2.** Internet: https://github.com/RVillarraso/Deep_Learning/blob/master/Inception%20ResNet-v2.png

[23] **Estructura del modelo Inception V4.** Internet: https://github.com/RVillarraso/Deep_Learning/blob/master/Inception-v4.png

[24] Chollet F. **Xception: Deep Learning with Depthwise Separable Convolutions.** Google Inc. http://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf [Acceso: Mayo 2018]

Software

[25] **Anaconda Navigator 3.** Internet: https://www.anaconda.com/download/

[26] **TensorFlow con Anaconda 3.** Internet: https://www.tensorflow.org/install/install_windows

[27] **Keras para TensorFlow.** Internet**:** https://keras.io/#installation

[28] **Pulse Secure Connect.** Internet: https://www.pulsesecure.net/accesssuite/

[29] **7-Zip Manager para Windows 64 bits.** Internet: https://www.7-zip.org/

[30] **Python 3.6.** Internet: https://www.python.org/downloads/release/python-360/

Análisis Económico

[31] **Microscopio Olympus BX43**. Internet: https://www.labx.com/item/olympus-microscope-bx43-with-tilting-head-and-2x-objective/LV37592361 [Acceso: Mayo, 2018]

[32] **Cámara OPTIKA C-P8 digital para microscopio**. Internet: https://www.microscopeinternational.com/product/optika-c-p8-8-3-megapixel-high-performance-cmos-camera/ [Acceso: Junio, 2018]

[33] **Tarjeta gráfica** NVIDA Titan Xp. Internet: https://www.nvidia.com/en-us/titan/titan-xp/ [Acceso: Junio, 2018]